Generating FDS Fire Simulation Input using IFC-based Building Information Model

By

Johannes A W Dimyadi

Supervised by

Dr. Michael Spearpoint

Dr. Robert Amor

Fire Engineering Research Report 07/1

March 2007

A research project report submitted in partial fulfilment of the requirements for the degree of Master of Engineering in Fire Engineering at the University of Canterbury.

School of Engineering, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
Phone +64 3 364 2250, Facsimile +64 3 364 2758
http://www.civil.canterbury.ac.nz/

For a full list of reports please visit http://www.civil.canterbury.ac.nz/fire/fe_resrch_reps.shtml

Acknowledgement

I would like to thank Dr. Michael Spearpoint and Dr. Robert Amor of the Department of Computer Science at the University of Auckland for their time, guidance and valuable advice during this project. I would also like to express my appreciation to Dr. Charley Fleischmann, Professor David Purser and Dr. Erica Dalziell for their wisdom and knowledge and for the inspiration they gave me during my study at the University of Canterbury. In addition, I would also like to acknowledge the New Zealand Fire Service Commission for their continued sponsorship of the Fire Engineering programme at the University of Canterbury.

I am grateful to my family, friends and colleagues for their encouragement.

Last but not least, thanks to Grace Wang for her support.

1. Dr Michael Spearpoint is a New Zealand Fire Service Commission senior lecturer in Fire Engineering at the University of Canterbury. His doctoral research was in the integration of building product models with fire simulation applications. Among other affiliations, he is a member of the Australasian Chapter of the International Alliance for Interoperability.

2. Dr. Robert Amor is an Associate Professor and Head of Department of the Department of Computer Science at the University of Auckland. Since 1985, his research interests have been in the application of information technology to Architecture, Engineering, Construction and Facility Management (AEC/FM) fields. He is currently one of the directors of the Australasian Chapter of the International Alliance for Interoperability.

Abstract

The adoption of a performance-based design approach in the regulatory framework and the accessibility of advanced computing technology have provided an incentive for fire engineers to use more sophisticated and computational intensive numerical simulation models such as Fire Dynamics Simulator (FDS) to solve complex fire engineering problems.

While other disciplines within the AEC/FM industry are starting to enjoy the benefits of sharing digital building model data, the current fire modelling practice still uses a somewhat conventional approach in gathering the basic building information. The conventional paper-centric practice represents a duplication of effort and contributes to errors as well as an unnecessarily high overhead in the fire simulation process. Sharing a common digital building information model using standard specifications such as the Industry Foundation Classes (IFC) can provide a considerable costs saving as well as an improvement on the quality and accuracy of the information exchanged.

The report looks at the current practice of CAD and building information modelling, highlights some of the constraints and the challenges in the data exchange process. The report also reviews an IFC based software parser for fire simulations and provides recommendations for extending the scope of its data mapping implementation. A web-based application has been developed specifically to demonstrate the extent in which IFC-based building information model can be used to generate the input data required for FDS simulations.

Contents

List of Figures

List of Tables

1.0 Introduction

1.1 Background

Fire Dynamics Simulator (FDS), a computational fluid dynamics (CFD) software application developed by the US National Institute of Standards and Technology (NIST), has become a useful tool since the adoption of a performance-based fire engineering design approach in the regulatory framework.

The availability and accessibility of faster computer processors with larger memory capacity makes it more practical for fire engineers to use such sophisticated and computational intensive numerical simulation tools to solve complex fire engineering problems.

However, the current fire modelling practice still uses a somewhat conventional paper-based approach in gathering basic building geometry information, which contributes to high overhead costs in the preparation for fire simulations. The conventional fire modelling process typically starts with the manual gathering of building geometry and other building information from either printed plans or electronic Computer Aided Design (CAD) drawing files. The information must then be transcribed, either manually or electronically, and used to construct a set of data and instructions in the format recognised by the fire simulation software applications. Each fire simulation software application may have different requirements on how to represent the building geometry. FDS, in particular, views a building as a series of orthogonal rectangular blocks or cuboids, each defined by a pair of bounding coordinates.

Although there are software tools available to assist with the creation of FDS input data, these tools are either only compatible with specific CAD output data formats, or they require manual data entry of the building geometry and topology information from scratch in a proprietary format. There are significant benefits in sharing a standardised digital representation of buildings in which common information can be mapped across various formats and for different input requirements of fire simulation applications.

A study by NIST in 2004 reported a $15.8 billion annual cost to the US capital facilities industry as a consequence of inadequate interoperability due to the “the highly fragmented nature of the industry, the industry’s continued paper-based business practices, a lack of standardization, and inconsistent technology adoption among stakeholders" [Gallaher, 2004].

Interoperability has been one of the major research and development areas in information technology in recent times. In the context of the architecture, engineering, construction, and facilities management (AEC/FM) industry, interoperability is the ability to exchange a standardised non-proprietary digital building information model. The Industry Foundation Classes (IFC) data model is one such model. IFC has now been endorsed as an international standard and has started to be adopted and utilised by various disciplines within the AEC/FM community.

1.2 IFC (Industry Foundation Classes)

Industry Foundation Classes (IFC) is an open standard object-oriented information model developed by the International Alliance for Interoperability (IAI), an international consortium of organisations in the AEC/FM industry. The IAI was formed in 1995 and consisted of regional chapters in nine different countries, i.e. North America, United Kingdom, Germany, France, Scandinavia, Japan, Singapore, Korea and Australia. Currently it has almost 500 members with 11 regional chapters in 24 countries. Its objective is to develop a universal standard for information sharing and interoperability of intelligent digital building models developed in object-based systems throughout all phases of the building life-cycle [IAI, 2006a].

The development of IFC model or specification began in May 1996 and was based on a number of earlier works for integrating building energy software applications. It was endorsed by the International Standards Organisation (ISO) in 2005 as a set of Publicly Available Specifications (PAS) under ISO/PAS 16739:2005 [Liebich, 2002]. The current official version of the model is IFC2x3 (IFC2x Edition 3), released in February 2006 [IAI, 2006a].

The model consists of a comprehensive set of AEC/FM industry related physical and functional objects definitions and specifications as well as a collection of extensible property sets, published as a set of HTML documents. It covers tangible entities (walls, windows, doors, pumps, etc.) as well as abstract entities (spaces, constraints, relationships, etc.). The shape representation used in the IFC model has been adopted from other international standards, namely ISO 10303-42:2003 [ISO, 2003] and ISO 10303-43:2000 [ISO, 2000].

IFC model is written in EXPRESS (ISO 10303-11:2004), which is a general purpose non-proprietary product data modelling and definition language very similar to Unified Modelling Language (UML) used by many commercial software applications.

IFC model data is currently exchanged between IFC-compliant software applications using STEP (Standard for Exchange of Product Data) Part 21 physical file (ISO 10303-21:2002), which was initiated by ISO in 1984 and developed by an international consortium of the aviation, automotive, ship-building, and related industries. Other available methods of exchanging IFC model information include a database access using the Standard Data Access Interface, the SDAI (ISO 10303-22:1998), as well as the more recent ifcXML (ISO 10303-28:2003] mapping [Liebich, 2005].

The IAI publishes other informational documents that are not part of ISO/PAS 16739, such as the IFC Specification Development Guide [Wix, 1998], the IFC Model Implementation Guide [Liebich, 2004], and the IFC Property Set Development Guide [Adachi, 2001].

1.3 Building Information Modelling (BIM)

CAD has been used as an electronic drafting tool for almost half a century. For the past decade, AEC/FM industry specific CAD technology has evolved into the new generation of building modelling concept currently known in the industry as Building Information Modelling (BIM) [IAI, 2006a]. Other terminologies have also been used or suggested by various individuals and organisations in the industry. This includes terms such as the Virtual Building (VB) [Graphisoft, 2003], Virtual Building Environment (VBE), Architectural Information Modelling (AIM) [Jensen, 2003], Parametric Building Modelling (PBM), Digital Building Modelling (DBM), Building Life-cycle Modelling (BLM) [Khemlani, 2004a], Building Life-cycle Management, etc.

BIM can be viewed as a collaborative object-oriented modelling extension to CAD. It integrates all information related to the building in one virtual information model that can be maintained throughout the whole life-cycle of the building (Refer Section 3.2).

The Facilities Information Council (FIC) of the US National Institute of Building Sciences (NIBS), which is a council to IAI-North American Chapter, recently formed a committee to create the US National Building Information Model Standard (NBIMS). This is currently being developed and expected to be completed and released by the end of 2006. The committee has been formed to “improve the performance of facilities over their full life-cycle by fostering a common, standard and integrated life-cycle information model for the AEC/FM industry” [FIC, 2006]. The working definition of BIM as originally developed by the Facilities Information Council (FIC) of NIBS is “A computable representation of the physical and functional characteristics of a facility and its related project/lifecycle information using open industry standards to inform business decision making for realising better value. BIM can integrate all the relevant aspects into a coherent organisation of data that computer applications can access, modify and/or add to, if authorised to do so” [FIC, 2004].

1.4 IFC interoperability and BuildingSMART

The IAI created a mission called BuildingSMART in 2005 to encourage a smarter process of managing the project life-cycle of the built environment by collaborative efforts and sharing of building information using the IFC-based BIM standards.

The industry’s effort to adopt the BuildingSMART approach has started with pioneering projects initiated by regulatory bodies and building authorities. For example, the Building and Construction Authority of the Republic of Singapore has started imposing the requirement for building designers to submit IFC-compliant electronic documents online for automated building code checking [BCA, 2005 and Lakshmi, 2005a]. In 2004, the UK Department of Trade and Industry commissioned the development of an IFC-based system to manage the operation and maintenance of buildings [ifc-mBomb, 2004].

It is envisioned that the BuildingSMART approach will be adopted in all phases of the project life-cycle including planning and design, construction, procurement, facility operations, major refurbishment and

demolition. The as-built information together with the entire valuable operational data is captured in one virtual model that can be preserved and used beyond the life of the project.

In a live presentation given at the IFC conference at Oslo, Norway on 1st June 2005, it was demonstrated that by using the BuildingSMART approach, 15 separate software applications could work concurrently with a single IFC building model. The applications ranged from terrain modelling, through geographic information system and architectural design to building services design [Stangeland, 2005].

1.5 FDS (Fire Dynamics Simulator)

FDS is a computational fluid dynamics (CFD) or field model tool of fire-driven fluid flow simulation. Field models typically divide the space of interest into a large number of discreet three-dimensional control volumes for which the fundamental equations governing the conservation of mass, species, temperature, velocity and density are solved.

FDS model numerically solves a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires [McGrattan, 2005a].

Although Direct Numerical Simulation (DNS) technique can be employed under sufficiently fine grid conditions, the default computational mode uses the Smagorinsky form of Large-Eddy Simulation (LES) technique to solve large scale hydrodynamic turbulence, a condition that typically occurs in fires. Sub-models are employed to deal with other fire related phenomena such as radiative and convective heat transfer, detector activation and sprinkler sprays.

There are three main sub-models currently used in FDS, i.e. Hydrodynamic model, Combustion model and Radiation Transport model [McGrattan, 2005a]. Revisions and improvements to the models are being incorporated into each new release of FDS.

The first official version of FDS was publicly released by NIST in 2000, although it existed previously in various forms since 1980 [McGrattan, 2005a]. The current version of FDS is Version 4.0.7, released on 23 March 2006.

FDS is a command-line software application written in Fortran 90 programming language without a graphical user interface and currently available in compiled versions for Windows as well as Unix/Linux operating systems. A copy of the software is made freely available from a dedicated web site maintained by NIST, i.e. http://fire.nist.gov/fds

FDS requires an input file for each fire scenario to be simulated. The input file is in a simple text format containing specially formatted lines of data. These lines of data may represent the building geometry, materials specification, computational scope, grid resolution, boundary conditions, design fire and energy source parameters, and specifications of fire safety and mechanical systems, as well as the simulation output types (refer Section 6.4).

The simulation outputs from FDS can be visualised graphically in an interactive 3D environment using a companion software application called SmokeView, which is also developed by NIST and written in C and Fortran 90 using OpenGL 3D graphics and GLUT/GLUI libraries [Forney, 2006].

SmokeView is also used as a visualisation tool by CFAST version 6.0, a zone-model fire simulation application developed by NIST [Peacock, 2005].

FDS also comes with a software utility that can be used to convert some of its output files to the comma separated values (csv) text format for other post-processing purposes.

Fire modelling using FDS is not the focus of this project and therefore has not been specifically addressed in this report. However, there have been a number of research works undertaken on various fire modelling aspects using FDS. Examples are FDS validation work by Clement [Clement, 2000], fire behaviour studies at the Centre of Environmental Safety and Risk Engineering at the Victoria University of Technology [Moghaddam, 2004], and the more recent numerical simulation work by Chiam [Chiam, 2005].

The generation, construction and placement of objects in FDS are described in detail in Chapter 4.0.

1.6 IfcSTEP Parser Software Application

ifcSTEP Parser is the successor to ifcXML Parser, a software application developed at the University of Canterbury in 2003 [Spearpoint, 2003]. The application has been developed to parse the STEP physical file of the IFC-based building model and to generate a relevant subset of fire engineering entities as an XML output document [Spearpoint, 2005]. This output document represents the intermediate data file, which can be used by software interfaces developed for creating specific input data required by various fire simulation applications.

XML is a text-based document format containing human-readable self-described data in a tree structure, which is popular for exchanging information across multiple platforms particularly over the internet. Similar to HTML, XML documents contain data elements and attributes enclosed in a pair of open and close tags. Unlike HTML, the tag names used in XML documents are generally user-defined (Figure 1).

```
<project id="#8716" name="TWOROOM2J-REVIT9">
  <units id="#8748" length="MILLIMETRE" />
</project>
<storey id="#1666" name="Level 1">
  <wall id="#12" name="WallA" length_x="3600" length_y="0" height="2550" pos_x="6297" pos_y="9691" pos_z="0">
    <materials id="#6541">
      <material id="#6540" description="Default Wall" thickness="100" />
    </materials>
  </wall>
...
  </storey>
```

Figure 1: Extract of an XML output document

The current version of ifcSTEP Parser is a command line processor without a graphical user interface and supports the IFC model up to version 2x2. The scope of IFC data mapping implementation in this version is quite limited, however it has been demonstrated that it can be used by a software interface, ifcSTEP-BRANZFIRE [Spearpoint, 2005] developed to generate the input data required by BRANZFIRE zone-model fire simulation application [Wade, 2003].

1.7 Research Objectives

This research project aims to achieve the following objectives:

1. To review the scope of IFC data mapping implementation in the current version of ifcSTEP Parser and to propose appropriate changes or extensions to meet the minimum mapping requirements for generating the input data for FDS.

2. To develop a software application as an interface to ifcSTEP Parser for parsing the XML output document and generating the input data required by FDS.

3. To construct a set of test case models using an industry standard BIM authoring software application for verifying the mapping implementation.

1.8 Report Structure and Outline

The first chapter of the report highlights the impetus for research, establishes the objectives, and provides some background information. It introduces the concept of IFC and BIM, as well as provides a brief description of FDS and ifcSTEP Parser software applications.

Subsequent chapters address IFC data model, BIM and FDS in greater detail and provide an in-depth analysis on the data mapping requirements between IFC building model, ifcSTEP Parser's XML intermediate file and FDS input data. The proposed extensions to ifcSTEP Parser and the development of a software interface for generating the FDS input data are also described and documented in detail.

These are followed by a chapter on the limitations of the project as well as a scope for potential future works.

The Appendices provide a list of test case models constructed and analysed in the project, snippets of program code and data files, as well as other supporting data and information.

1.9 Keywords, Terminologies and Definitions

A list of abbreviations, acronyms, keywords and phrases used in the report can be found in Appendix 1. These are mostly terminologies specific to the information technology domain, which may not be familiar to the reader. Others are keywords intended to have specific meaning in the context of this project. Many of the definitions provided are taken or adapted from the articles in the online wiki encyclopaedia as well as other references as appropriate.

2.0 IFC and STEP

2.1 IFC Data Model

Since its initial release, the IFC data model has progressively been improved to incorporate more aspects of buildings and project information. The current IFC 2x3 version of the model contains 653 entities and over 300 supplementary data types as well as extensible property sets [Liebich, 2006]. The additions and revised entities in the latest version of IFC do not have any impact on the objects exchanged in this project. As ifcSTEP Parser currently only supports IFC 2x2 and there are no obvious advantages to upgrade, the data and mapping analyses have been based on this earlier version.

The IFC data model is essentially constructed in a hierarchical structure. However, its object-oriented design allows inheritance among entities and complex relationships to exist between entities.

IFC entities have the naming convention with an '*Ifc*' prefix. For example: *IfcWallStandardCase*, *IfcOpeningElement*, *IfcDoor*, *IfcWindow*, etc. Similar naming convention is given to the property sets, but with a *Pset* prefix. The naming convention is indicated by italicised texts where used in the body of this report.

The IFC model structure is often represented using the EXPRESS-G notation, a graphical modelling language subset of EXPRESS developed within STEP used for identifying model classes, data attributes and their relationships. Figure 2 shows part of IFC model structure in EXPRESS-G showing main building elements (other details omitted for clarity).

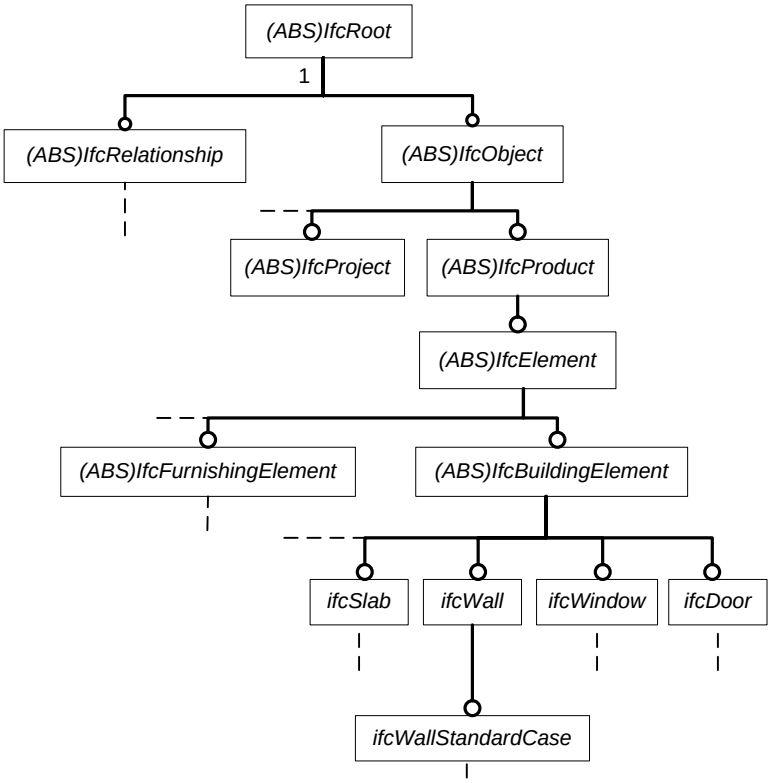


Figure 2: EXPRESS-G diagram showing part of the IFC model structure

2.2 STEP Physical File (SPF)

The most common method of IFC model information exchange between the compliant software applications is currently the STEP physical file (SPF), which is in text format containing human readable ASCII characters. STEP is defined in ISO 10303-21:2002 Clear Text Encoding of the Exchange Structure.

Each STEP file consists of two main sections, namely the Header and Data sections. The Header section contains several groups of information relating to the file itself. The Data section contains sequentially numbered lines of application data [Figure 3].

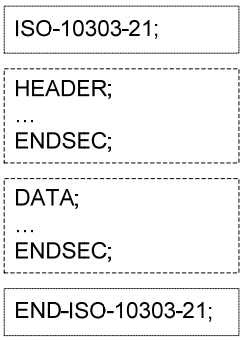


Figure 3: STEP file structure

Each line of data contains instances of entities, properties, cross-references, attributes, type definitions, or mapped attribute values. The line numbers are used to identify the data, provide cross referencing to related data, and describe the relationship between related data in the hierarchy. An example of STEP file is shown in Figure 4.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('IFC2X_PLATFORM'),'2;1');
FILE_NAME('MultiRoom2','2006-09-08T21:58:26',(''),(''),'Autodesk Revit Building 9 - 1.0','20060505_2300','');
FILE_SCHEMA(('IFC2X2_FINAL'));
ENDSEC;
DATA;
#1=IFCORGANIZATION($,'Autodesk Revit Building 9',$,$,$);
#2=IFCAPPLICATION(#1,'Release 9','Autodesk Revit Building 9','Revit');
#4=IFCDIRECTION((0.,0.,1.));
#5=IFCSIUNIT(*,.LENGTHUNIT.,.MILLI.,.METRE.);
#6=IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#7=IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#8=IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.01745329251994328),#6);
#9=IFCCONVERSIONBASEDUNIT(#7,.PLANEANGLEUNIT.,'DEGREE',#8);
#10=IFCSIUNIT(*,.TIMEUNIT.,$,.SECOND.);
#11=IFCUNITASSIGNMENT((#5,#9,#10));
```

Figure 4: An extract from a STEP physical file (SPF)

The size of a STEP file is usually quite large. For example, a simple two room test model would contain as many as 10,000 lines of data in the STEP file when exported using Autodesk's Revit Building 9 software package.

Due to the current variations in the modelling standards and IFC export implementations adopted by different BIM authoring software packages, the same building model may be exchanged slightly differently in the STEP files. Parser software applications must be able to cope with these variations and interpret the model correctly as intended.

The observations and analyses of the IFC data and mapping carried out in this project are based on the BIM and IFC implementation in the Autodesk's Revit Building 9 software package and the corresponding STEP files.

2.3 Accessing IFC Model Data

There are a number of IFC compatible software tools available to support the analysis and display of IFC based models and to assist the development of IFC based applications (Table 1). A list of IFC and BIM resources including links to software applications download sites is provided in the Appendix 7. A comprehensive list of IFC compatible software tools can also be found on the IAI-International website [IAI, 2006b].

Application Developer	**Application Name**	**Category**
DDS	DDS IFC Viewer	Model viewer
Forschungszentrum Karlsruhe	IfcStoreyViewer	Model viewer
Forschungszentrum Karlsruhe	IfcObjectCounter	Model checker
GEM Team Solutions	IFCQuickBrowser	Model data browser
Institute for System Programming, Russian Academy of Sciences	IFC/VRML translator	Data converter
NIST	CIS/2 to IFC Translator	Data converter
Octaga	Octaga Modeller	Model viewer
Secom Co. Ltd	SECOM model server	ActiveX component
Solibri	Solibri Model Checker (SMC)	Model checker
Solibri	Solibri Model Optimizer	Model Optimizer
TNO	IFC Engine	Model browser/viewer

Table 1: IFC supported software tools

2.3.1 IFCStoreyViewer

IfcStoreyViewer is a stand-alone software utility developed by Forschungszentrum Karlsruhe (Karlsruhe Research Centre), a European science and engineering research institution funded jointly by the Federal Republic of Germany and the State of Baden-Württemberg. It has been developed as a research tool and is made freely available, hence the reason for the choice.

IfcStoreyViewer is intended to be used for parsing STEP files of BIM models containing one or more storeys. It can be used to view the models graphically and interactively or to query the entities and to display their properties.

This software utility has been used in this project to observe the impact of various model changes on the data exchanged. Figure 5 shows a screenshot of the user interface displaying one of the test case models.

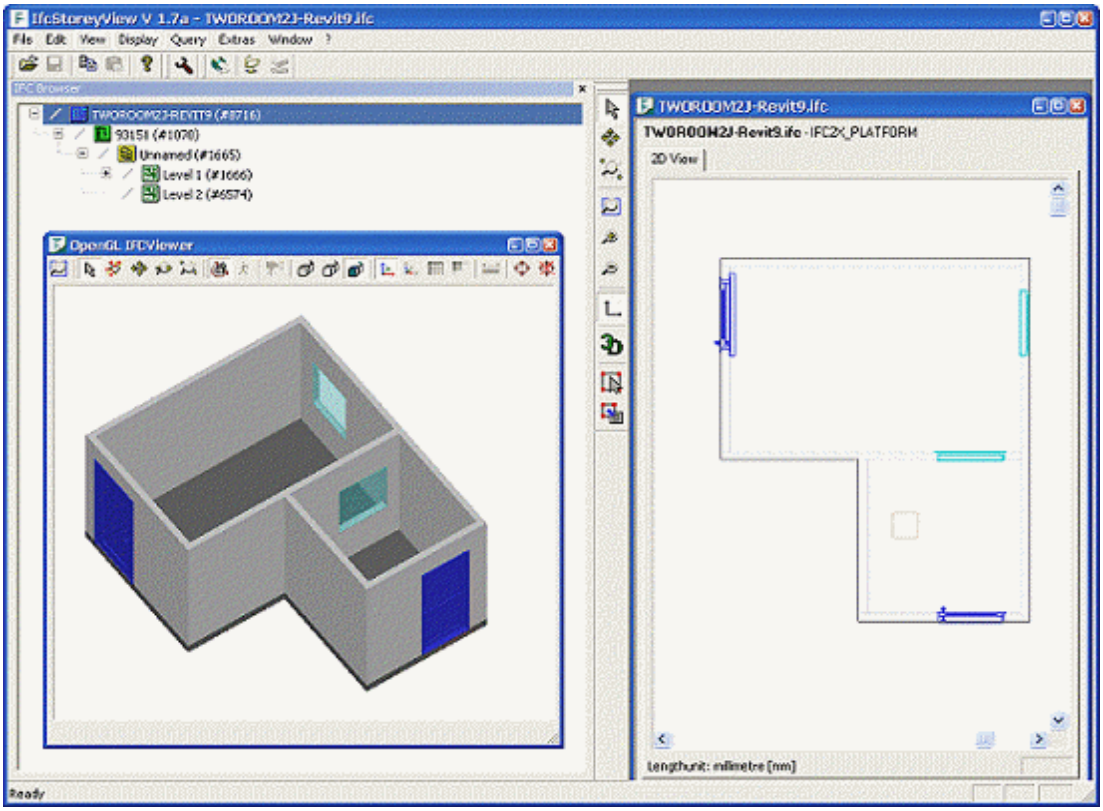


Figure 5: A screenshot of the IfcStoreyViewer environment

2.3.2 IfcQuickBrowser

IfcQuickBrowser is a software utility developed by G.E.M. Team Solutions, the software developer that produced the IFC 2x export interface for the Autodesk's Architectural Desktop software package. It is a reasonably compact utility with a very intuitive interface. The fact that it is also freely available makes it even a better choice for use in the project.

Figure 6 shows the user interface of ifcQuickBrowser displaying a part of the STEP file data tree for one of the test case models.

IfcQuickBrowser can be used to navigate through the STEP file and display the contents in the tree structure. For each line of data in the tree containing a branch, the utility searches through the whole file and display the related lower level 'child' data on that branch. On demand, it continues to display each level of child data down the hierarchy until the last branch.

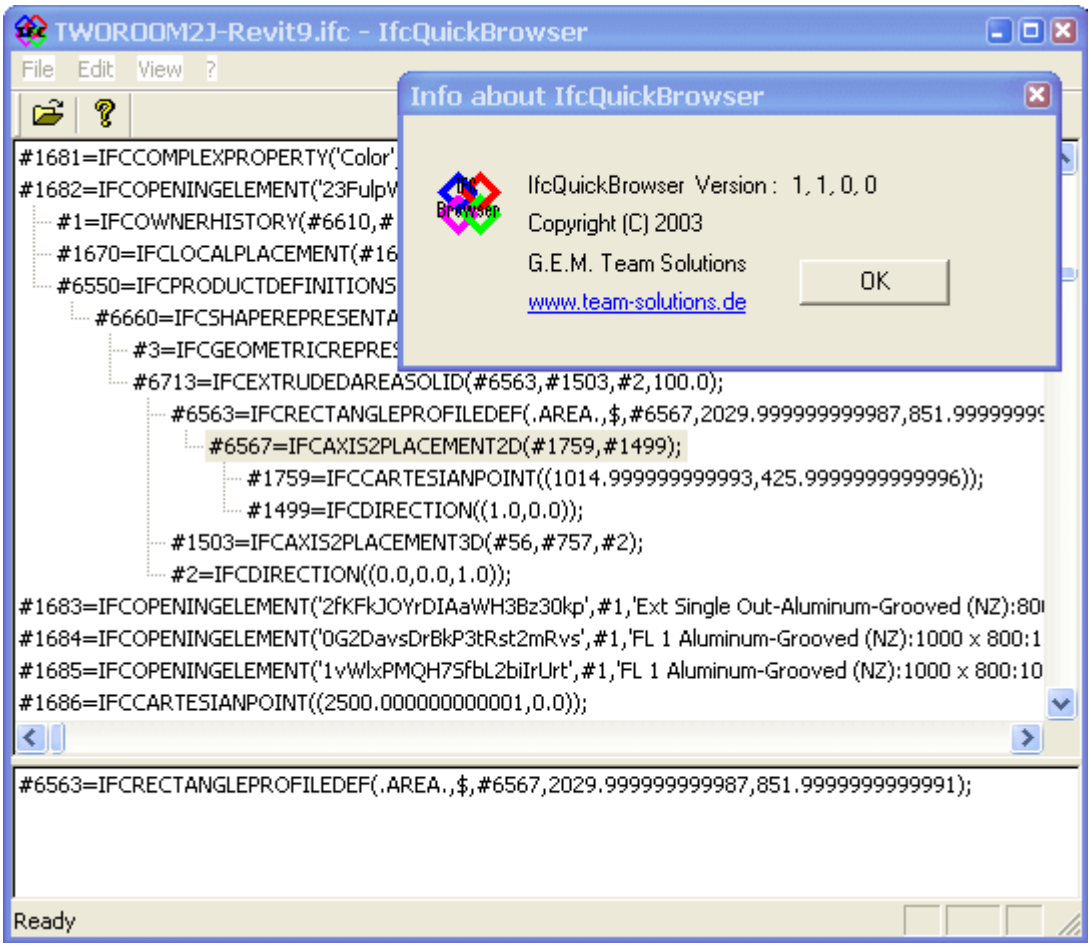


Figure 6: A screenshot of a STEP file in IfcQuickBrowser

Inversely, given a child data on a lower level branch, the corresponding higher level parent data can also be displayed. As shown in Figure 6, the element #6567 *IfcAxis2Placement2D* is being selected and the branch expanded with two child data, i.e. elements #1759 and #1499 being shown. In the lower pane, the corresponding parent data immediately above the selected element in the tree is also displayed, i.e. element #6563.

IfcQuickBrowser also has a generic search function that can be used to search for any characters or texts contained in the STEP file.

The software utility has been used in this project as a tool for browsing and navigating STEP files. It has also been used to provide a means of observing the effects on the data exchange as the result of changes made in the source model.

3.0 CAD and BIM

3.1 CAD

Computer Aided Design (CAD) has a history of almost half of the century. It was initially developed to assist with the design and fabrication of machinery parts, tools and components in the automotive and aerospace industry. The adoption of CAD by the AEC/FM industry became widespread following the release of commercial CAD products on PC computers in the early 1980s.

Conventional CAD software packages can be categorised into 2D drafting systems and 3D geometric modellers. 2D CAD is effectively an electronic drafting tool. It fundamentally involves creating and editing of vector-based entities consisting of lines and arcs on a two-dimensional plane. 3D CAD technology generally employs either surface or solid modelling techniques, manually or parameter-driven, in a three-dimensional virtual environment. 3D CAD deals with the creation, construction, assembly and manipulation of geometrical objects from basic cuboids, prisms, cylinders, spheres to polyhedrons and more complex primitives.

There are several major conventional CAD products that have been used as electronic drafting systems in the AEC/FM industry over the years. Among the more popular ones are Autodesk's AutoCAD [Walker, 1994], Bentley's MicroStation [Davis, 2002], and Graphisoft's ArchiCAD [Lakshmi, 2006].

3.1.1 Surface Modelling

3D surface modelling is most useful in the creation of open freeform surfaces and objects with complex shapes, but this technique can also be used in some CAD software applications to construct basic geometrical objects with flat or faceted surfaces.

Surface modelling may be used to create 3D geometrically closed objects by defining the shell or the boundary surfaces enclosing the volume of the objects. The technique focuses on the accurate geometric representation of the object rather than concerns about its physical properties. The equivalent technique used in solid modelling is known as Boundary Representation (BREP) (refer Section 3.1.2).

For example, a flat surface in a shape of a polygon in 3D space has a set of coordinates defining its vertices (Figure 7). There are a number of different standards of 3D flat surface entities recognised and in use by CAD applications, namely POLYFACE, REGION and 3DFACE.

As indicated in Figure 7, a rectangular block representing a building element such as a wall must be defined by six flat surfaces, which may be represented by 3DFACE entities. Each 3DFACE entity has four sides or boundaries as well as four sets of coordinates defining its vertices.

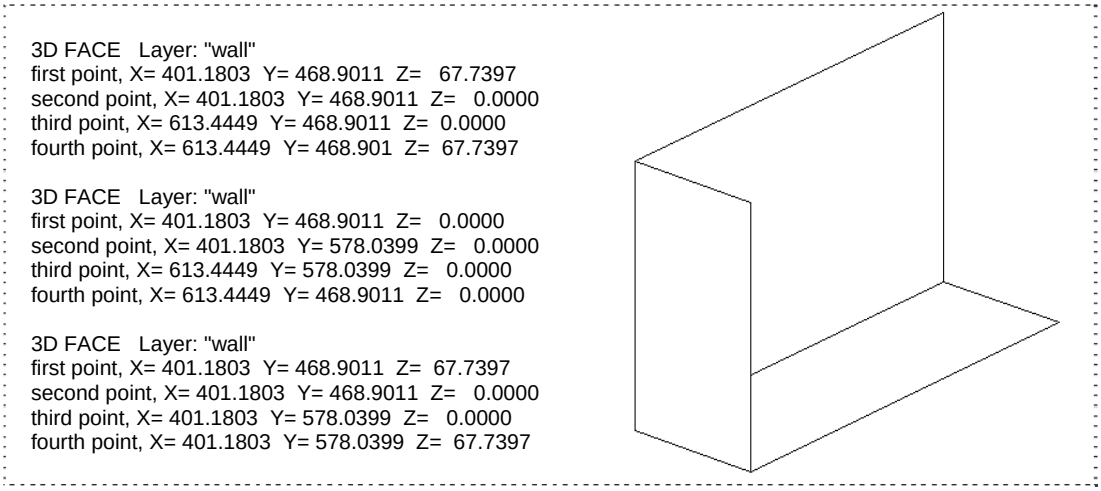


Figure 7: Coordinates of 3DFACE entities extracted from AutoCAD

3.1.2 Solid Modelling

Solid modelling may be a more appropriate technique in representing 3D objects in certain applications particularly when the objects' physical properties are relevant. The two most common techniques used in solid modelling are Constructive Solid Geometry (CSG) and Boundary Representation (BREP). In CSG, a solid object representing a building element such as a wall is normally constructed by the extrusion of a rectangular surface object or region in either x, y or z axes. Editing of solids is achieved by Boolean operations such as union, subtract, and intersect, etc. with other solid primitives. BREP represents a solid by a series of connected boundary surface elements, which are defined by vertexes, edges and loops.

There are several solid modelling software engines available that are commonly incorporated into commercial CAD applications. Among these engines, 3D ACIS modeller is one of the most popular engines used by major CAD applications including AutoCAD. ACIS engine produces 3DSOLID object type entities (Figure 8) as a CSG or a BREP.

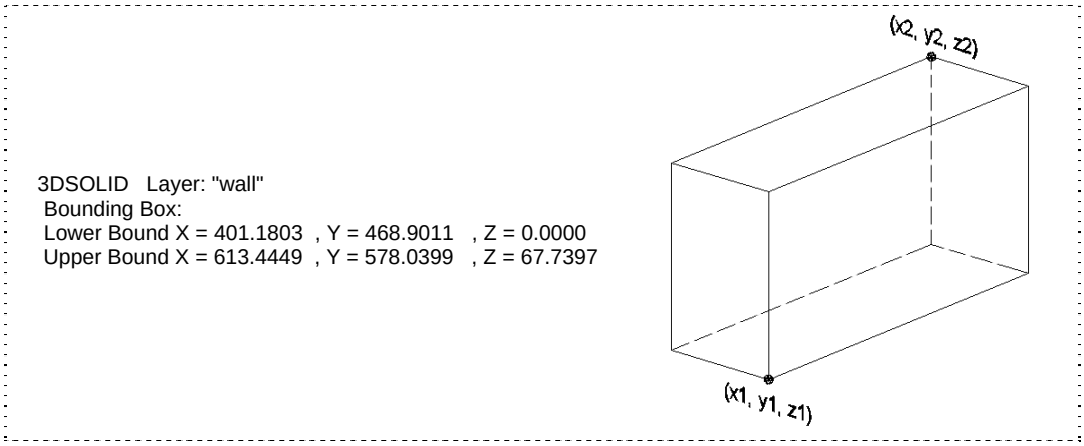


Figure 8: A 3DSOLID entity extracted from AutoCAD

A 3DSOLID entity representing a rectangular block such as that shown in Figure 8 is geometrically defined by a pair of lower and upper bounding coordinates.

3.1.3 CAD data exchange

CAD data generally consists of 2D and 3D entities representing points, lines, arcs, polygons, circles, text as well as regions and solid objects. Advanced CAD geometrical objects may also contain limited set of attributes and data with user-defined values.

Files produced by commercial applications such as Autodesk's AutoCAD are generally in proprietary formats, although common formats such as DXF and IGES have been introduced to allow data exchange between different versions of the application as well as with external applications [Amor, 1997].

The DXF format has been used to exchange 2D building layout as image overlays onto which other data and information are superimposed. Examples of application using this DXF data exchange method include Simulex [Thompson, 1995] for evacuation simulations, SMARTFIRE [Frost, 2001] for fire simulations, Aperture Visual Information Manager [Wikipedia contributors, 2006] for facilities management, as well as FDS via a conversion utility called DXF2FDS (refer Section 4.5.1). The use of DXF, however, often requires that the source data extracted from the building model contains certain compatible types of entity and are constructed in specific manners to be interpreted correctly. For example, Simulex would not interpret walls correctly if they are exchanged as polylines rather than separate line entities.

There have also been other third-party software utilities, add-ons and plug-ins available to allow CAD data to be accessed and exchanged by external applications. The data exchange process generally involves retrieving predefined attribute values of entities and exporting them to a database table or a spreadsheet or direct linking them with the external applications. This method of data sharing has been used in HVAC and fire sprinklers design packages such as the Elite Software Suites [http://www.elitesoft.com], and for quantity take-offs and costs estimating such as the AutoAids Estimator by Salesoft CAD Solutions [http://www.salesoft.co.nz].

For the past decade, more data-aware features have been incorporated into major CAD products such as the Autodesk's AutoCAD. However, these conventional CAD applications, by design and by tradition, lend themselves for use more as the electronic drafting tool for the production of drawings as the end-product of the design process.

3.2 BIM

The concept of BIM differs from the conventional CAD in the approach to the design process. BIM puts an emphasis on creating and managing a richer set of project information in one consolidated data repository representing the virtual building model that can be shared by all the project stakeholders. The graphical representation is merely one aspect of BIM.

As advanced 3D graphics technology is readily available, CAD applications have naturally transformed into BIM applications packaged with both the 3D graphical representation as well as the data-aware

object-oriented modelling features. Using the BIM approach to design, drawings become the by-product of the design process. Current BIM applications can automatically produce drawings, reports and schedules as part of the modelling process. Any changes and additions to the model are immediately and automatically reflected on all of the drawings, reports and schedules.

In a typical BIM application environment, every virtual components and objects created or assembled can contain a set of properties that make them aware of their placement and their relationships with each other in the overall model. One of the advanced features in the current BIM applications is the nD modelling technique, which can be used to extend 3D building information to incorporate multi-dimensional aspects of the project. For example, in 4D modelling, objects are given phase definitions to make them aware of their existence in time. BIM allows the multi-aspects of the design information to be incorporated into one virtual building model that are accessible at every stage throughout the life-cycle of the building. One of the potential advantages of BIM is the ability to exchange and share digital building models containing the rich multi-dimensional project information using a common data exchange protocol such as the IFC model.

Applications such as code-compliance checking, energy analysis, structural analysis and fire simulations are beginning to benefit from the one-way sharing of the IFC based BIM model.

Although IFC-based BIM data sharing has started to be used by a few downstream applications, the collaborative potential of BIM is yet to be fully realised. Currently, only a relatively small number of industry stakeholders have invested in BIM applications and there are still some incompatibility issues between BIM products. In this transitional period, BIM applications are being viewed as the extension to CAD, which focuses on the generation of drawings and documentations as the primary output of design.

It appears that the implementation of a full two-way interoperability is currently still a challenge. This could be partly due to the variations in the implementation between BIM products and also the challenge in mapping certain information representations having different level of granularity as required by different applications.

Most of the current BIM authoring applications lack built-in modelling constraints. However, independent software tools such as Solibri Model Checker can be used to check the model integrity against the IFC specifications and highlight any errors or inconsistencies so that it can be corrected before the data is exchanged [Khemlani, 2006].

3.3 BIM authoring applications

Currently there are only a few BIM authoring applications that have been certified by IAI as being compatible for exchanging IFC model data. Table 2 shows a sample list of the available IFC certified commercial BIM authoring products. Most of the products listed have the import and export support for the latest version of the IFC model, but some certified for export only and others are only compatible with an earlier version of the IFC model. In the absence of built-in IFC export features, independent software conversion tools are often available. Alternatively, compatible BIM models may be imported into one of the industry standard software packages, edited as necessary and then exported to IFC.

Commercially available BIM authoring applications now available in three main categories, each tailored to suit one of the three major domains of the industry, i.e. architectural, structural and building services. Autodesk's Revit Building, Graphisoft's ArchiCAD and Nemetschek's Allplan Architecture are products intended for the architectural and general buildings design. Bentley System' Microstation V8 is marketed for infrastructure design. Autodesk's Revit Structure, Tekla's Tekla Structures and Nemetscheck's Allplan Engineering are developed for the design and modelling of structures with the emphasis on civil and structural engineering. Autodesk's Revit Systems and DDS's Building Services Partner suite are designed for the modelling of buildings from the building services or building systems perspective.

Developer	**Product Name**	**IFC support**
Autodesk	Architectural Desktop 2007	IFC 2x
Autodesk	Revit Building, Revit Structure, Revit Systems	IFC 2x
Bentley System	Microstation V8	IFC 2x
Data Design System (DDS)	Building House Partner, Electrical Partner, HVAC Partner	IFC 2x
Graphisoft	ArchiCAD	IFC 2x
Microsoft	Visio 2000	IFC 2.0
Nemetscheck	Allplan Architecture, Allplan Engineering, Allplan Building Services, Allplan Precast Concrete.	IFC 2x
Tekla	Tekla Structures	IFC 2x
Olog Granlund	SMOG	IFC 1.5.1
Gehry Technology	Digital Project	-

Table 2: BIM authoring applications

Some examples of buildings designed using BIM approach are shown in Table 3.

Building Project	**Authoring application**
300 m tall 90 storey Eureka Tower, Residential Apartment Melbourne, Australia	ArchiCAD
HUT-600 extension, Helsinki University of Technology, Finland	ArchiCAD
42 storey, mixed-use complex Mondrian Tower, San Diego, USA	Revit
12-storey, mixed-use complex, Axiom Tower, San Diego, USA	Revit
Ford Centre of Engineering and Manufacturing Excellence (CEME) Dagenham, UK	MicroStation
9 storey mixed-use new Westminster theatre complex Luxury apartment, retails and theatre, London, UK	MicroStation

Table 3: Sample buildings designed with BIM approach

Different BIM authoring software packages have different strengths and weaknesses and some work better in certain types of project than others. Design firms, particularly those involved with multi-disciplinary projects, often use more than one type of BIM software packages to suit different project requirements [Lakshmi, 2005b].

3.3.1 Autodesk's Revit Building software package

Autodesk's Revit Building 9 software package has been chosen as the BIM authoring tool to construct the test case models used in this project (refer to Chapter 7.0 and Appendix 2). One of the reasons for the choice was that an academic licence was available at a considerable discount. Apart from that, it was considered worthwhile using another industry's leading BIM product as an alternative to ArchiCAD, which has been used previously in a similar work [Spearpoint, 2005].

The IFC conforming STEP files of the models are obtained by a direct export from Revit Building. The IFC export function of the software package caters for three different IFC formats, namely IFC2x2, IFC2x3, and IFC BCA ePlan Check. The latter is intended for the ePlan submission required by the Building and Construction Authority of the Republic of Singapore.

The Revit Building software package has an intuitive user interface and a simple modelling environment that makes it reasonably easy to use. The version used in this project also comes with a comprehensive set of standard objects and components relevant to the New Zealand AEC/FM industry.

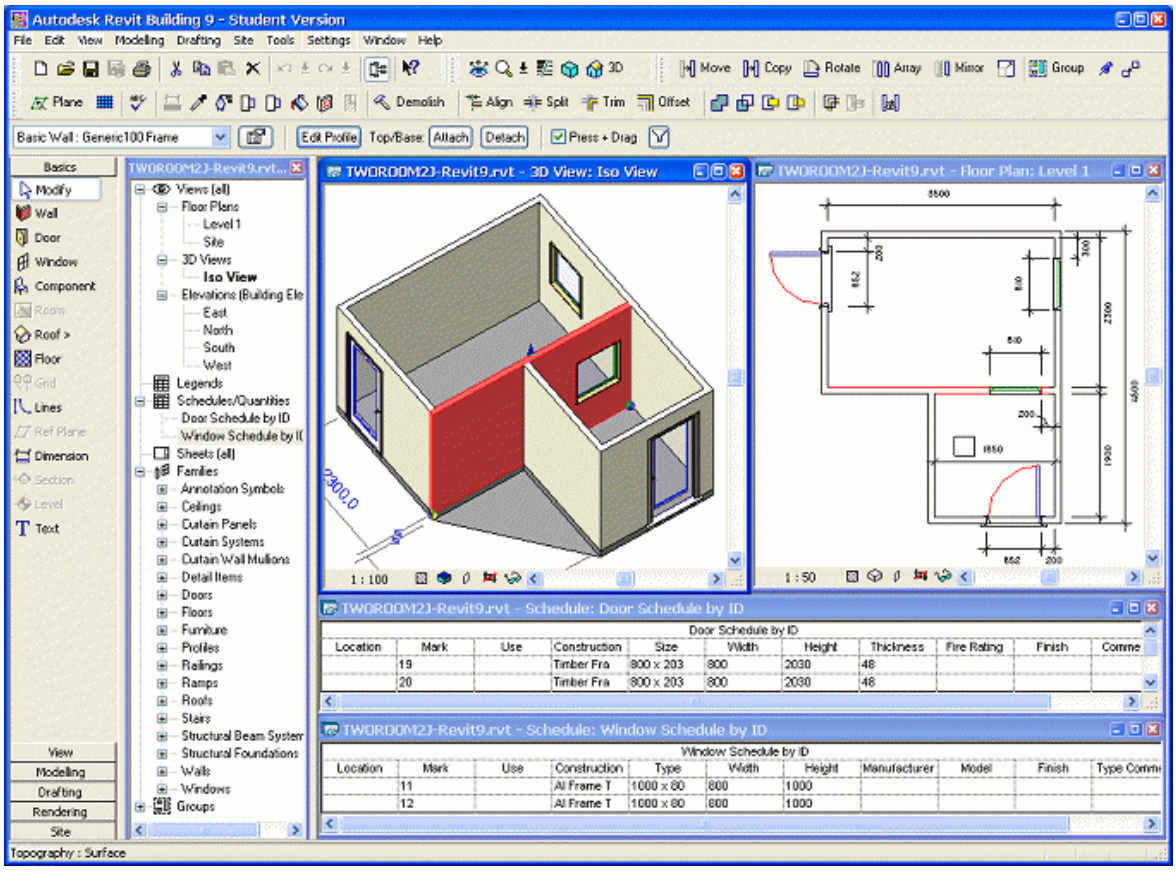


Figure 9: Autodesk's Revit Building 9 Modelling Environment

Figure 9 shows the modelling environment in Revit Building. The wall element containing an internal window highlighted in Figure 9 is shown as being selected for query. The query displays a set of properties for the highlighted wall as shown in Figure 10.

Each instance of objects or components created in the model has “Instance Parameters” defining its placement, dimensions, functions and phase in the project. Components such as a wall have user-defined “Type Parameters” consisting of the construction properties, graphics properties, and identity data such as the name of the manufacturer, fire rating and cost.

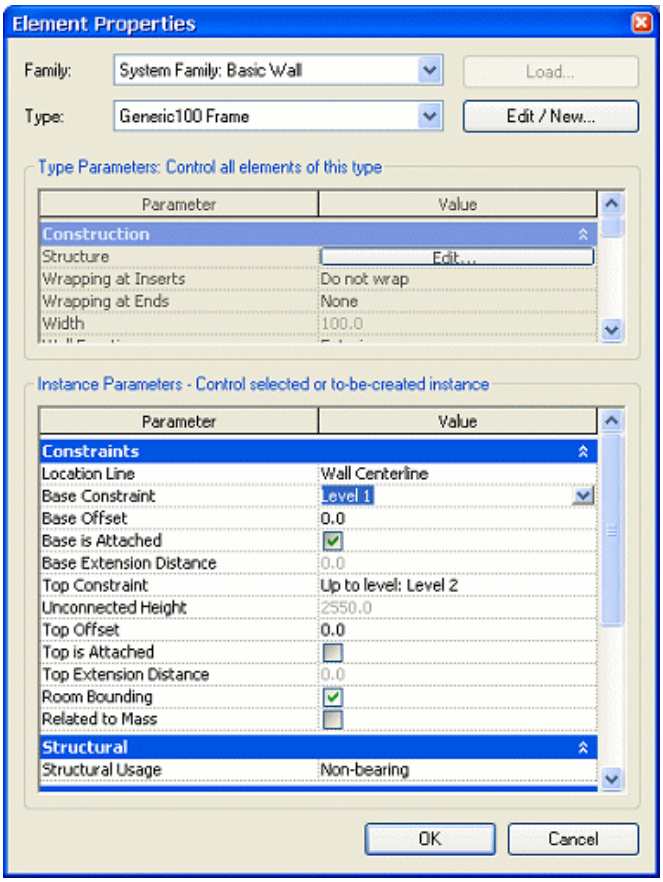


Figure 10: Selected wall properties in Revit Building

In Revit Building, the modelling process usually starts with the placement of a selected type of wall having a pre-defined height on the current working plane followed by the placement of doors and windows on the wall by dragging and dropping the selected component from the list of component families.

Once placed, building components can be visually moved by editing the appropriate dimension values or by physically dragging the components. Their properties can also be changed by querying and editing.

The model can be displayed in plan view, elevations, or in the interactive 3D mode. Any other views such as sections and details can also be generated automatically by placing the section markers at the required location.

3.4 BIM standards and constraints

Downstream applications have specific requirements and often only concern with certain aspects of the building. Therefore, they only need to focus on a relevant subset of the IFC representation of the building for data access efficiency.

BIM authoring applications, by nature, allow some degrees of flexibility in the modelling techniques and object representations. Furthermore, BIM authoring applications may have slightly different IFC export implementations. However, there are several ways to ensure that any applications sharing the model information can interpret the data accurately, as follows:

1. The data mapping methods used by the downstream applications must be able to cope with the varying cases of representation obtained as the result of different modelling techniques used or different export implementations adopted by the authoring tools.
2. There needs to be a set of standards to be followed in the modelling process. This standard must include data sharing and collaboration protocols. One such standard might be the upcoming US NBIMS that is currently under development [FIC, 2006].
3. Using a domain specific independent parser developed specifically to cope with the variations in the model exchanged and to produce a consistent set of intermediate data to be exchanged by the downstream application.

Some of the more common variants produced as the result of different modelling practice or techniques used with authoring applications may include the followings:

1. A model may or may not include entities required by the downstream applications. For example, a topographical surface must be specifically created in Revit Building in order for *IfcSite* entity to be exchanged by the STEP file. Similarly, *IfcSpace* entity would be exchanged only if a space entity exists.
2. The local placement of the wall maybe on the outer face of the wall, the inner face of the wall, or on the centreline of the wall.
3. Walls may be constructed with arbitrary directions in random orders, e.g. in the positive or negative X or Y axis directions.
4. Walls may have different types of joint at corners, i.e. mitred or butt jointed.
5. A wall opening may be created in different ways, i.e. by constructing three segments of walls around the opening, by cutting a hole in the wall using one of the modelling tools, or by placing a door frame without the door leaf on the wall.
6. The type of construction may determine how the wall is exchanged and what type of wall it is exchanged as.

3.5 BIM for downstream applications

There are currently a number of downstream commercial applications in the AEC/FM industry that are already sharing the IFC building model either directly or indirectly via various conversion utilities. Examples of these applications are given in Table 4.

Developer	Application Name	Category
Active Facility [ActiveFacility, 2005]	Active Facility	BIM web services for FM
Green Building Studio inc.	Green Building Studio	Energy simulation
IES	Virtual Environment (VE)	Environmental simulations
Innovaya	Innovaya	Quantity & cost Estimating
ISO 10301	CIMSteel Integration Standard (CIS/2)	Structural Steel Product Model
MRO Software	Maximo 5	Asset Management
NIST	CONTAMW	Indoor Air Quality Analysis
novaSPRINT	Corenet e-Plan Check System	Code-checking
Olof-Granlund Oy	BSPro COM-Server	Data access middleware
Olog Granlund Oy	Riuska	Energy simulation
US Department of Energy	EnergyPlus	Energy Simulation
Vizelia	Facility on line	FM application

Table 4: Example of downstream applications sharing BIM data

Details of these applications can be obtained from their respective developers' web sites (refer Appendix 7].

3.6 BIM for fire engineering applications

The principal designer of a building is usually an architect, a building engineer, a structural engineer, or an architectural designer. It is very seldom that a fire engineer or a building services engineer assumes the role of the principal designer of the building.

Conventionally in the fire engineering scene, a fire engineer would be given a set of printed drawings or CAD files from the principal designer of the building. The required information would then be manually collated or extracted from these sources for use in the code compliance assessments, for carrying out fire engineering analysis or for conducting fire simulations.

The availability of a BIM model would mean that a fire engineer may now obtain a copy of the IFC output file in SPF format instead of the paper drawings or conventional CAD files. There are a number of scenarios in which this new form of information could be used in fire engineering applications:

1. The STEP file can be imported into a BIM authoring/editing application in which the model can be studied in detail in preparation for a preliminary assessment of the building from the fire engineering perspective.

2. Using the modelling or editing tools in the BIM application, the fire engineer may be able to make simplifications to the building or add certain elements and properties to the model in order to create a fire scenario to be analysed or simulated. The result may be exported as a revised STEP file, which represents the fire scenario to be analysed or simulated.

3. The revised STEP file may be parsed by a fire engineering specific tool such as the ifcSTEP Parser and to generate the input data required by the fire simulation software applications using software interfaces developed specifically for this purpose.

4. The generated input data for the fire simulations can further be edited, if required, using any dedicated software interfaces developed specifically for the fire simulation application.

5. Any recommended modifications or additions to the building as the result of the fire engineering assessment may be highlighted or incorporated into the model using a BIM authoring tool. For example, the fire rating of a door may be changed in the model, or new fire engineering properties may be added to the model as appropriate. The result may be communicated to the principal designer in the form of a STEP file, from which the revised elements or new properties could be extracted and incorporated into the master building model.

Some commonly used fire engineering applications that can potentially benefit from sharing BIM model are listed in Table 5.

Developer	**Application Name**	**Category**
BRANZ	BRANZFire	Zone model fire simulation
Fire Safety Engineering Group (FSEG)	SMARTFIRE	CFD model fire simulation
Fire Safety Engineering Group (FSEG)	Exodus	Evacuation simulation
IES Virtual Environment	Simulex	Evacuation simulation
NIST	FDS	CFD model fire simulation
NIST	CFAST	Zone model fire simulation

Table 5: Example of fire engineering applications

4.0 Fire Dynamics Simulator (FDS)

4.1 FDS Input Data

The input data for FDS consists of a set of namelist formatted lines of data in a simple text file representing a fire scenario to be simulated (refer to Figure 54 for an example of the generated input file of a test case model). Each line of data begins with the name of a namelist group (HEAD, GRID, VENT, OBST, etc) prefixed by an ampersand (&) character. This is followed by a delimited list of parameters corresponding to that group. Each line of data is terminated with a forward slash (/) character (Figure 11) [McGrattan, 2005a].

All other non conforming texts in the input file are treated as comments or remarks and would not be processed by FDS.

```
&OBST XB=x1, x2, y1, y2, z1, z2, SURF_ID='PAINTED_GIB' / Wall ID=#53
&HOLE XB=x1, x2, y1, y2, z1, z2  / Door ID=#849
```

Figure 11: Sample lines of FDS input data

4.2 Building Element Representation in FDS

Building enclosure elements and solid objects are specified in the FDS input file as a series of orthogonal solid rectangular blocks or cuboids (or right parallelepipeds) representing flow obstacles, whereas doors and windows are viewed as holes allowing fluid and particles to flow through. As each solid rectangular obstruction blocks must be orthogonal with respect to each other, an inclined or diagonal wall or roof must be modelled in a stair-stepping manner conforming to the grid cells.

4.2.1 A Solid Block

For the &OBST (obstruction) group of input data, the parameters required are the sextuplet of coordinates defining the lower and upper bound of the cuboids, followed by the surface type and material definition. Refer to Figure 11 and Figure 12.

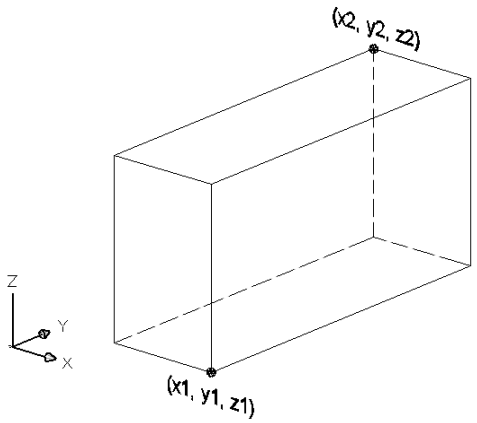


Figure 12: A solid obstruction block or a void representation in FDS

4.2.2 A Void in a Solid Block

A void in an obstruction block is specified with &HOLE namelist group having similar parameters to the &OBST group (Figure 11 and Figure 12). Any solid grid cells within the volume specified by the sextuplet on the &HOLE line are removed and the obstructions intersecting the volume are automatically broken up into smaller blocks (Figure 13).

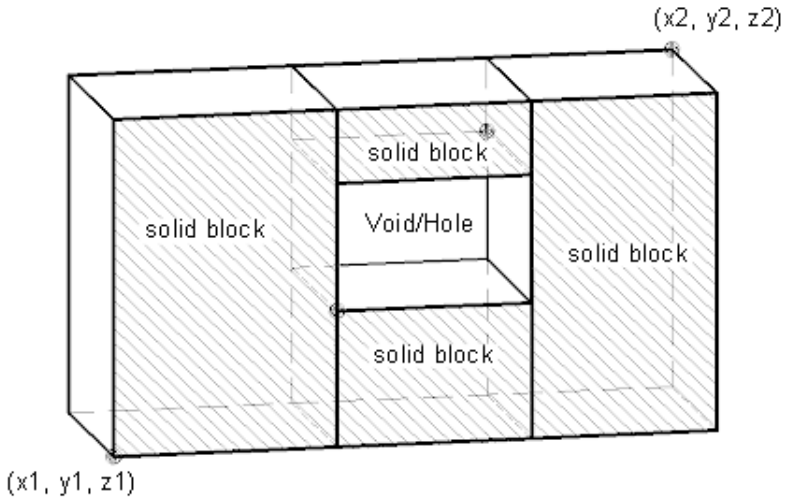


Figure 13: A void in a solid block created with &HOLE

T_CREATE and T_REMOVE parameters can be used in conjunction with &HOLE to specify the time in seconds at which the voids are to be created or removed, simulating the opening and closing of doors or windows. Specifying T_REMOVE = 0 is equivalent to filling up the hole, simulating a door or window that is to be kept closed for the duration of the simulation.

4.3 FDS Physical and Computational Domain

The FDS computational domain is user defined and usually represents the overall physical bounding box enclosing the building. However, it can be extended beyond the building boundaries

The computational domain is made up of one or more rectangular meshes, each with its own three-dimensional rectilinear grid system. The resolution of the grid is also user-specified in the input file, but the grid dimensions must meet the criteria for the Poisson solver, i.e. it must be in the form $2^l 3^m 5^n$, where l, m and n are integers [McGrattan, 2005b]. All solid obstructions and voids are forced to conform to the numerical grids.

The exterior boundary planes of the computational domain or the sides of the bounding box can be defined with &VENT namelist group as ‘OPEN’, ‘INERT’ or having a surface material applied to the plane.

In ifcSTEP-FDS, the entire side boundary planes have been defined as ‘OPEN’, which effectively means that they are flow transparent to the outside of the computational domain except for the areas obstructed by the external walls.

The bottom boundary plane has conveniently been defined as having ‘CARPET’ surface material. Similarly, the top boundary plane has been defined to have ‘PAINTED_GIB’ surface material (Figure 14).

```
****** DOMAIN BOUNDARIES
&VENT CB='XBAR0', SURF_ID='OPEN' /
&VENT CB='XBAR', SURF_ID='OPEN' /
&VENT CB='YBAR0', SURF_ID='OPEN' /
&VENT CB='YBAR', SURF_ID='OPEN' /
&VENT XB=6.297,9.997,7.241,9.741, 0.00, 0.00, SURF_ID='CARPET' / CARPET FLOOR
&VENT CB='ZBAR', SURF_ID='PAINTED_GIB' / PAINTED GIB CEILING
```

Figure 14: FDS Domain Boundaries definitions

T_OPEN and T_CLOSE parameters can be used in conjunction with &VENT to specify the time in seconds at which the area defined on the plane is to be opened or closed, hence simulating the opening or closing of windows and doors on or adjacent to the exterior boundary of the computational domain.

4.4 FDS Output

FDS output types are user specified in the input file prior to the start of each simulation. FDS can produce graphical output files containing the 3D model geometry, animated quantities per unit time (i.e. isosurface, slice or boundary files) as well as static pictures of the flow field in Plot3D format, which can be accessed and visualised using SmokeView. A number of csv files are also produced as the result of the simulation, which contain the time-history plots of various predicted quantities.

The type of output quantities that can be predicted by FDS include the space and boundary conditions such as the temperature profile, heat flux and mass flow rate, the history of heat release rates, the levels of oxygen, carbon dioxide and carbon monoxide, and other combustion products as specified.

Figure 15 shows a SmokeView snapshot of the space temperature profile (indicated by a colour spectrum) at a particular time in the FDS simulation of a test case model.

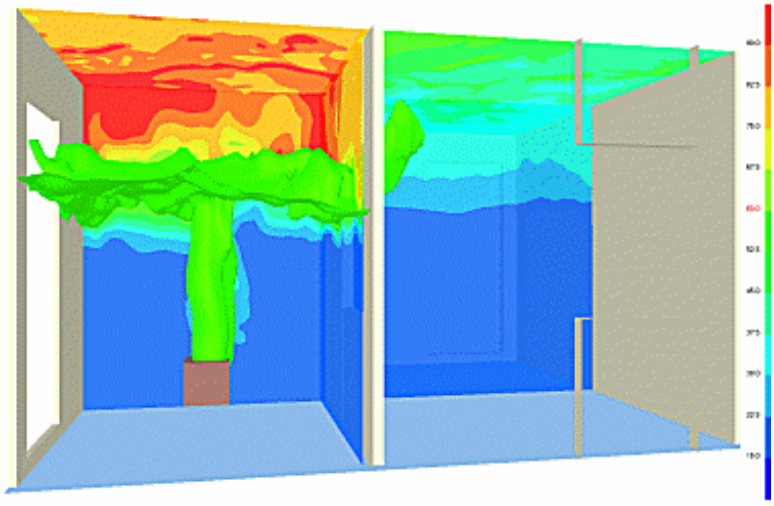


Figure 15: Space temperature snapshot of a test case model

FDS generates a set of output data files during each simulation. At the start of each simulation, a SmokeView output file is first created containing the building geometry visualisation data. The duration of the simulation can be set to zero initially, i.e. &TIME TWFIN=0.0 /, which prevents any computations from taking place except for the generation of the building geometry data. This would enable the building model specified in the input file to be previewed and verified before any intensive and time-consuming computations taking place.

4.5 Creating FDS Input Data

The creation of the FDS input file is a part of the fire modelling process and requires varying degrees of manual input and editing particularly when considering multiple fire scenarios.

Generally, the most time consuming part of the input data creation is the transfer of building geometry information from paper or CAD drawings to the format required by FDS. At the most laborious level, all the coordinates for the obstruction blocks representing the building geometry are manually determined by measurements and calculations from printed drawings. This process is particularly inefficient and error prone.

There are a number of software tools currently available to assist with the creation of the FDS input data, particularly with respect to the transfer of the building geometry and topology information. All of these tools require the reconstruction of the building in one form or another, using information derived from printed plans or CAD files. It is believed that there have not been any successful attempts in the industry for the development of a software tool that utilises the IFC data model to generate the FDS input data, although NIST has indicated the intention to explore the possibility of undertaking such a project [AIA Building Connections].

4.5.1 DXF2FDS using 3DFACE

DXF2FDS is a software conversion application developed using Microsoft .Net framework by Dr. Dave Sheppard, a research engineer for the US Bureau of Alcohol, Tobacco, Firearms, and Explosives (ATF). The application was written during 2001 when he was a visiting researcher at the US Building Fire Research Laboratory (BFRL) [NIST, 2006].

DXF2FDS reads a DXF output created by a specific CAD application and generates the FDS equivalent obstruction blocks representing the building geometry. It can also incorporate a set of pre-defined FDS input parameters such as the computational domain and grid sizes, selected surface materials, prescribed energy source, etc.

DXF2FDS has a limitation in that it only reads the 3DFACE entities and ignore all others from the DXF input file. A model constructed using other solid modelling techniques often cannot easily be converted directly into 3DFACE entities. Therefore, a tedious conversion process is often required or, alternatively, the model needs to be reconstructed using 3DFACE entities exclusively.

4.5.2 DXF to FDS tool using 3DSOLID

In 2004, a software utility application was written by the author [Dimyadi, 2004] to directly utilise the 3DSOLID entities produced by AutoCAD software package in generating the obstruction blocks data for the FDS (Figure 16).

The input to this software utility is a log file in a simple text format generated by the AutoCAD's object listing query and contains all 3DSOLID objects selected on a particular layer (Figure 17). The software utility reads the log file and picks up the bounding box coordinates of these entities to generate the obstruction blocks for the FDS.

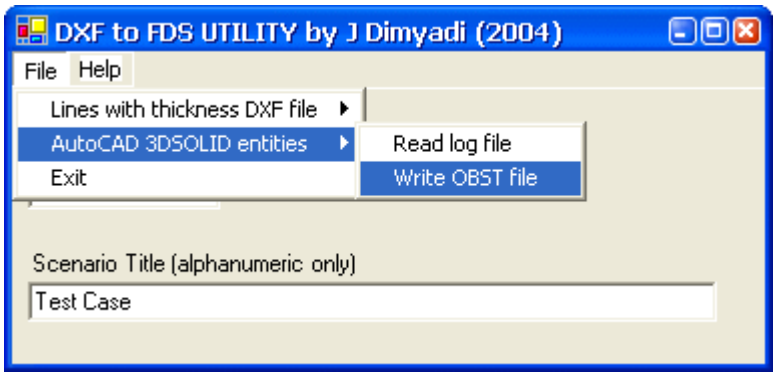


Figure 16: User Interface of DXF to FDS software utility

```
[ AutoCAD - Sat Oct 30 17:19:47 2004  ] ----------------------------------------

Select objects: ALL
66 found

3DSOLID   Layer: "wall"
Space: Model space
Handle = 298
Bounding Box:
Lower Bound X = 22.2000  , Y = 1.5000   , Z = 0.0000
Upper Bound X = 22.4000  , Y = 4.5000   , Z = 6.0000

3DSOLID   Layer: "wall"
Space: Model space
Handle = 297
Bounding Box:
Lower Bound X = 19.7000  , Y = 1.5000   , Z = 0.0000
Upper Bound X = 22.2000  , Y = 1.7000   , Z = 6.0000

3DSOLID   Layer: "wall"
Space: Model space
Handle = 294
Bounding Box:
Lower Bound X = 10.2000  , Y = 1.5000   , Z = 0.0000
Upper Bound X = 10.4000  , Y = 4.5000   , Z = 6.0000
```

Figure 17: Extract from an AutoCAD log file

The prerequisite of using this software tool is that the solid model of the building elements must first be constructed and assembled in the AutoCAD environment using a specific modelling convention and layering standards [Dimyadi, 2004]. More importantly, the building geometry information required to construct the solid models must first be manually derived from some printed plans or electronic files, which is inefficient as it represents a duplication of effort.

4.5.3 PyroSim Software Package

More recently, a commercial software application called PyroSim [Thunderhead Engineering Consultants, 2006] was developed and released by Thunderhead Engineering in the US with the support of a Small Business Innovation Research (SBIR) grant from the US National Science Foundation. This is a stand-alone graphical user interface (GUI) software application that can be used to construct, read and edit FDS input files.

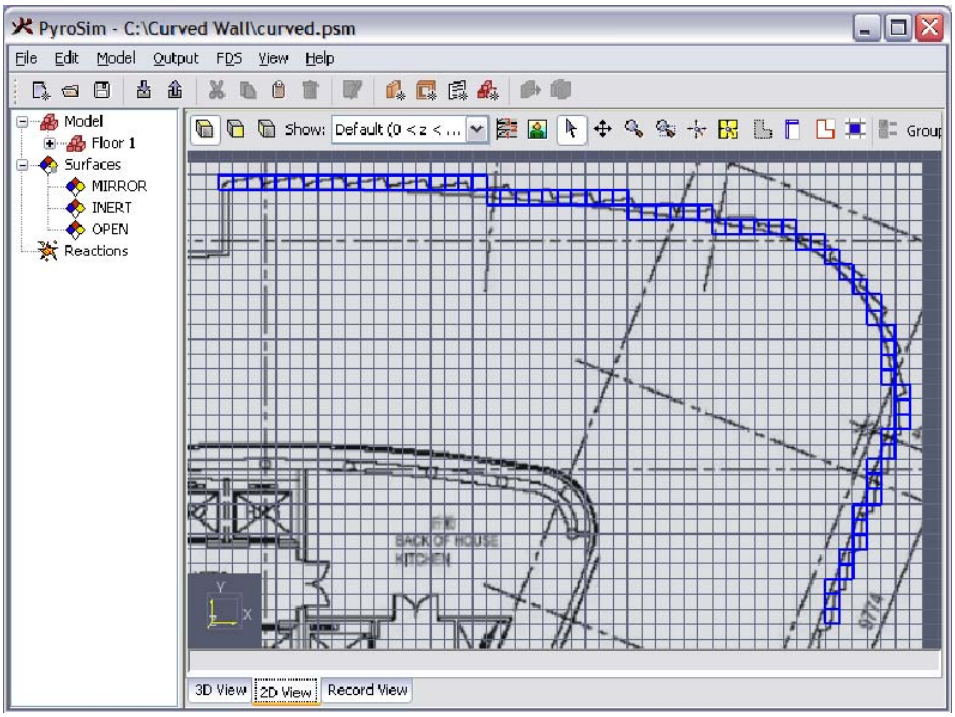


Figure 18: A screen-shot of the graphics editor of PyroSim

To assist with the construction of the 3D building model similar to the obstruction blocks in FDS, a 2D image of the building plan can be overlaid on the graphics editor screen to allow three-dimensional wall elements to be manually positioned by tracing over the lines (Figure 18). Stair-stepping is automatically applied to diagonal or curve walls to conform to the numerical grid system.

PyroSim has many useful features to enable the visual creation and editing of FDS input files and supports all of the input namelist groups available in the current version of FDS. It also allows FDS and Smokeview to be directly executed from within the application.

The current version of PyroSim does not support IFC import or export directly. However, it can be used to import the FDS input file generated by ifcSTEP-FDS for further editing before running the simulation. Figure 19 shows the generated FDS input file for a test case model being edited in PyroSim.

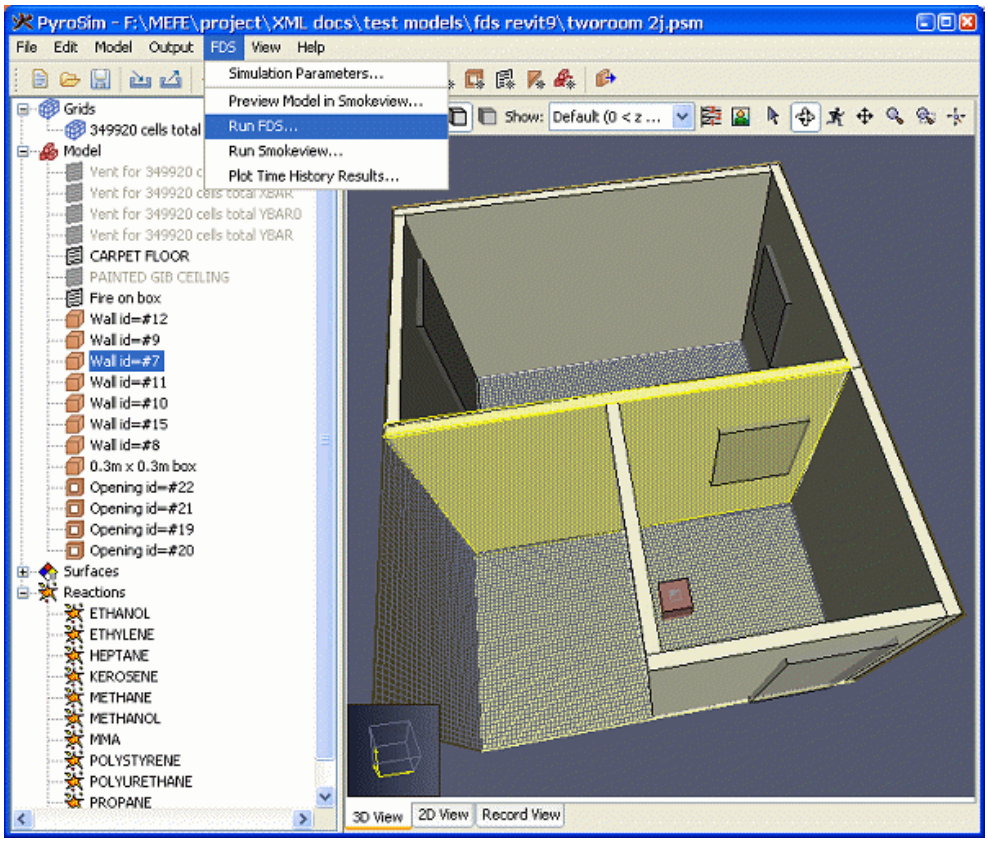


Figure 19: FDS input file for a test case model being edited in Pyrosim

4.5.4 Other Proprietary Software Tools

As FDS gains popularity among fire engineers, a number of in-house developed tools have emerged among consulting engineering practices to facilitate the creation of the FDS input data. Among these are stand-alone software utilities to create stair-stepping of diagonal, inclined or curved walls, and also various FDS obstruction blocks generators.

These proprietary software tools, however, do not have the capability of importing standard building information models, but instead still require varying degrees of manual input of the building geometry.

5.0 ifcSTEP Parser

The idea behind having a dedicated IFC parser for the fire-engineering domain is mainly to improve data access efficiency and maintain a consistent IFC version support. Software interfaces developed for various fire simulation applications only need to access the most relevant subset of IFC entities provided by the ifcSTEP Parser rather than processing the whole IFC model directly. The other advantage is that any revisions to the IFC model and modifications to the IFC entities only need to be implemented on the ifcSTEP Parser, which maintains a consistent common output file for use by all the software interfaces. The data exchange process from BIM to the fire simulation application is illustrated as shown in Figure 20 [adapted from Spearpoint, 2005]. Additionally, ifcSTEP Parser can be used as a repository of the fire engineering domain specific property sets to ensure that any supported property sets incorporated into the BIM models are mapped correctly to the target fire simulations software application.

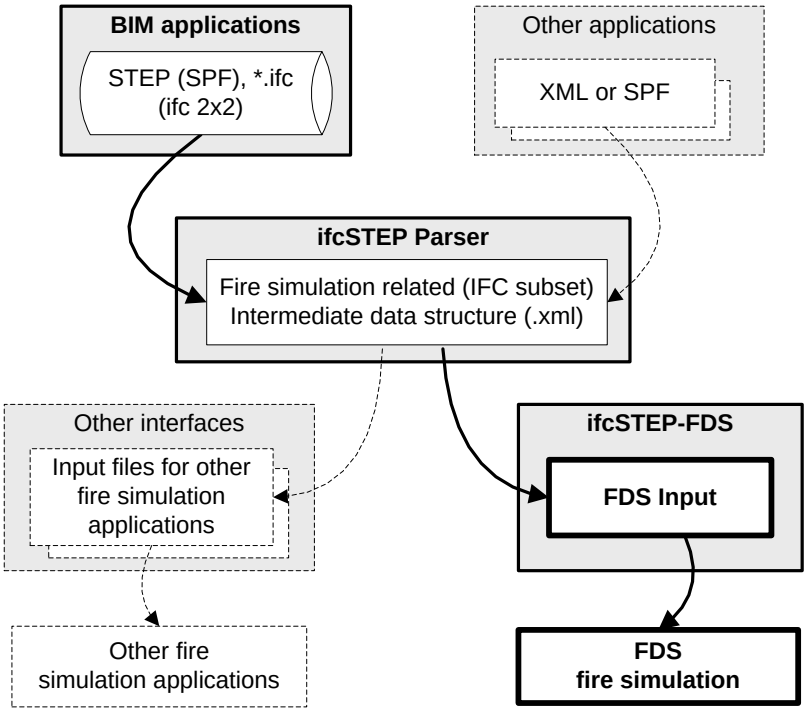


Figure 20: Data Exchange Process (adapted from [Spearpoint, 2005])

5.1 Extending ifcSTEP Parser

The scope of review and subsequent proposed extensions is limited to the geometric representation mapping of walls, wall openings, door and window openings, as well as the furniture elements placement.

5.1.1 Initial Scope of ifcSTEP Parser

At the outset of this project, the scope of the IFC data mapping implementation in ifcSTEP Parser is limited to providing a set of building elements (i.e. walls, doors, windows, openings, slab, etc) with dimensions and relative placement origins as exchanged by the ArchiCAD software package, which are

sufficient for the purposes of generating the BRANZFIRE input data as it does not concern with absolute positioning [Spearpoint, 2005]. However, objects' absolute positioning information is required by FDS, particularly with respect to the walls and wall openings.

The main limitations of the initial version of ifcSTEP Parser for the purposes of generating the FDS input data can be summarised as follows:

1. The direction of walls, door, window, and wall openings were not mapped. This would be essential in determining the absolute placement coordinates of the entities.

2. Placement coordinates were mapped directly from the STEP file without modifications for aligning with the standard reference axis. A convention would be necessary to accommodate for different cases due to the variants in modelling and implementation.

3. Only *IfcWallStandardCase* entities were supported.

4. Apart from the *IfcSite* and *IfcBuildingStorey* entities, the *IfcSpace* entity was expected in the STEP file, which was required by ifcSTEP-BRANZFIRE [Spearpoint, 2005].

IfcSTEP Parser exchanges information using the attributes for each relevant IFC entity in its XML output document. Each of the IFC entities is identified by a unique id number in the STEP file. The same sets of id numbers have been adopted by ifcSTEP Parser to identify the IFC entity mapped in the XML output document (Figure 21).

The scope of attributes attached to the wall and opening entities can be seen in an example XML document output of the initial version of ifcSTEP Parser (Figure 21).

```
<window id="#1299" name="$" width="620" soffit="1200" sill="900" pos_x="2080" pos_y="3140" pos_z="900" />
<door id="#603" name="$" width="900" height="2100" pos_x="13180" pos_y="3100" pos_z="0" />
<door id="#938" name="$" width="900" height="2100" pos_x="4397" pos_y="5410" pos_z="0" />
<door id="#1238" name="$" width="900" height="2100" pos_x="200" pos_y="3040" pos_z="0" />
<wall id="#1219" name="" pos_x="0" pos_y="3070" pos_z="0">
  <materials id="#1198">
    <material id="#1197" description="Double 1:8" thickness="70" />
  </materials>
</wall>
```

Figure 21: Part of an XML document output from ifcSTEP Parser

The initial scope of attributes attached to the wall, door and window opening entities are illustrated in Figure 22, Figure 23, and Figure 24, respectively.

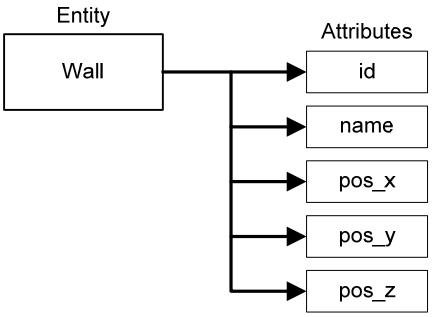


Figure 22: Initial attributes for walls

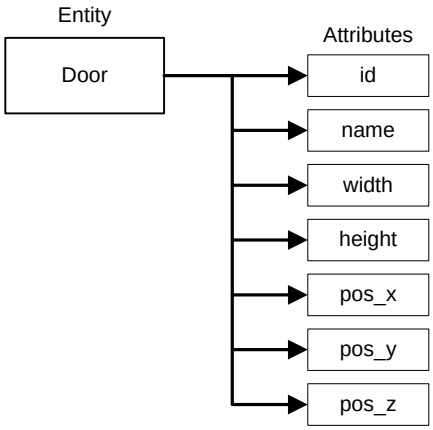


Figure 23: Initial attributes for doors

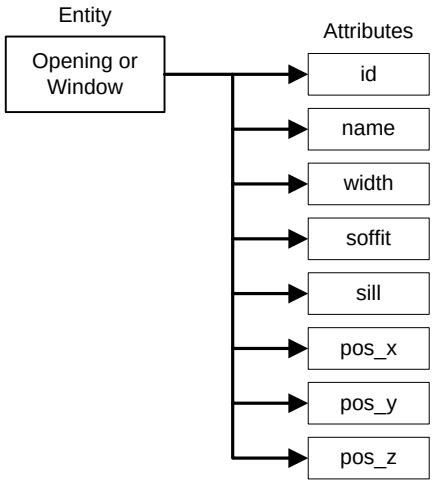


Figure 24: Initial attributes for windows or plain openings

5.1.2 Proposed Extensions

To represent a building for a fire simulation, BRANZFIRE is more concern with the volume of the spaces and the area of the openings than the absolute positioning of the walls and openings. On the other hand, FDS requires that all building elements are placed in the absolute positions. Therefore, a wider scope of information is required for mapping the building geometry to the representation expected by FDS.

To extend the scope of the information exchanged, some changes and additions to the mapping implementation in ifcSTEP Parser are required, as follows:

1. Establish a convention for the placement of walls and wall openings by referencing the placement coordinates on the wall axis, which is at the centreline of the wall.

2. Establish a convention for the placement of wall openings by realigning the opening direction with that of the parent wall and transforming the placement coordinates accordingly.

3. Revise the mapping method used for the placement of wall openings using the reference axis.

4. Add mapping of wall and wall opening directions.
5. Suggest methods to determine wall heights for non-standard walls.
6. Add mapping of furniture elements placement.

The revised implementation provides a new set of attributes to be added to the wall and wall opening entities in the output XML document, as illustrated in Figure 25, Figure 26, Figure 27.

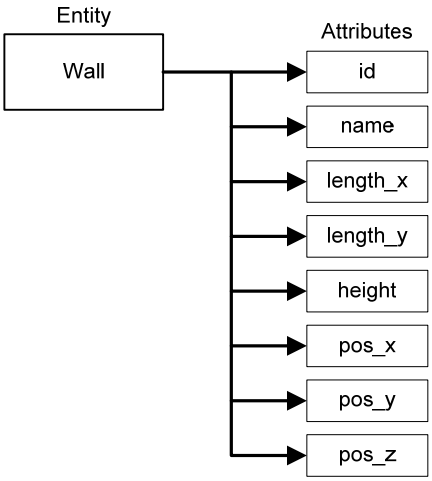


Figure 25: Extended scope of wall attributes

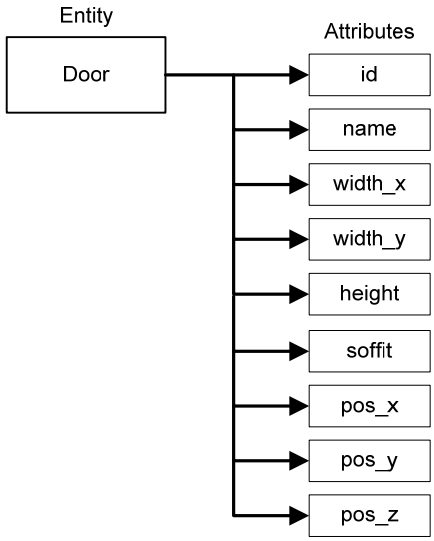


Figure 26: Extended scope of door attributes

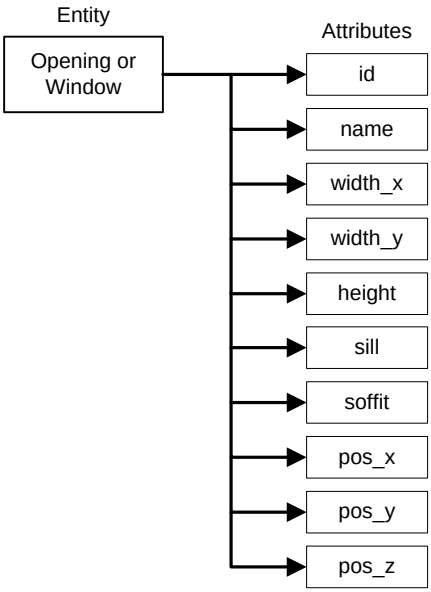


Figure 27: Extended scope of window/plain opening attributes

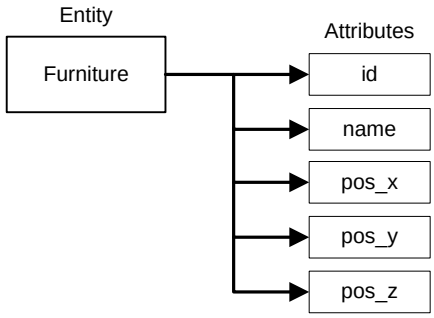


Figure 28: Extended scope of furniture placement

The data mapping analysis and the proposed implementation changes are described in detail in the following chapter of the report.

These recommended changes and extensions have all been implemented during the project and verified using the test case models (refer Chapter 7.0 and Appendix 2).

5.2 Mapping IFC Wall entities

5.2.1 Walls Representation

In the IFC model version 2x2, wall entities are exchanged in STEP as instances of *IfcWall* or *IfcWallStandardCase*.

The *IfcWallStandardCase* entity type supports standard walls, which have the following characteristics:

- Uniform overall thickness along the wall path.
- Wall path follows a straight line or a circular arc.
- Various offsets from the wall path may exist.
- Single or multiple material layers.
- Constant or a varying height along the wall path

All other cases are considered non-standard and categorised as specific wall type, which is supported by the *IfcWall* entity.

For the purposes of this project, the scope of *IfcWallStandardCase* and *IfcWall* entities have been restricted to the following:

- Uniform thickness along the wall path.
- Wall path follows a straight line with no offsets and parallel to either the X or Y axis, hence orthogonal with respect to each other.
- Wall consists of a single uniform material layer.
- Walls have a constant height throughout the building storey.
- In the presence of an *IfcWall* entity, walls are taken to have a constant height throughout the building storey.
- Wall vertical axis is perpendicular to the floor and ceiling planes.
- Wall base is at a constant elevation throughout the building storey.

Both standard and specific wall types have at least two shape representations in the IFC model, namely the wall axis and the wall body. Standard walls are typically defined by the centreline axis and the sectional profile.

The wall axis provides a reference for the material layer offset and also defines the wall path. It is also used as a reference axis for the placement of walls and wall openings (Figure 29). As shown in the illustration, the placement origins of the walls are expressed in the coordinate system of the building storey.

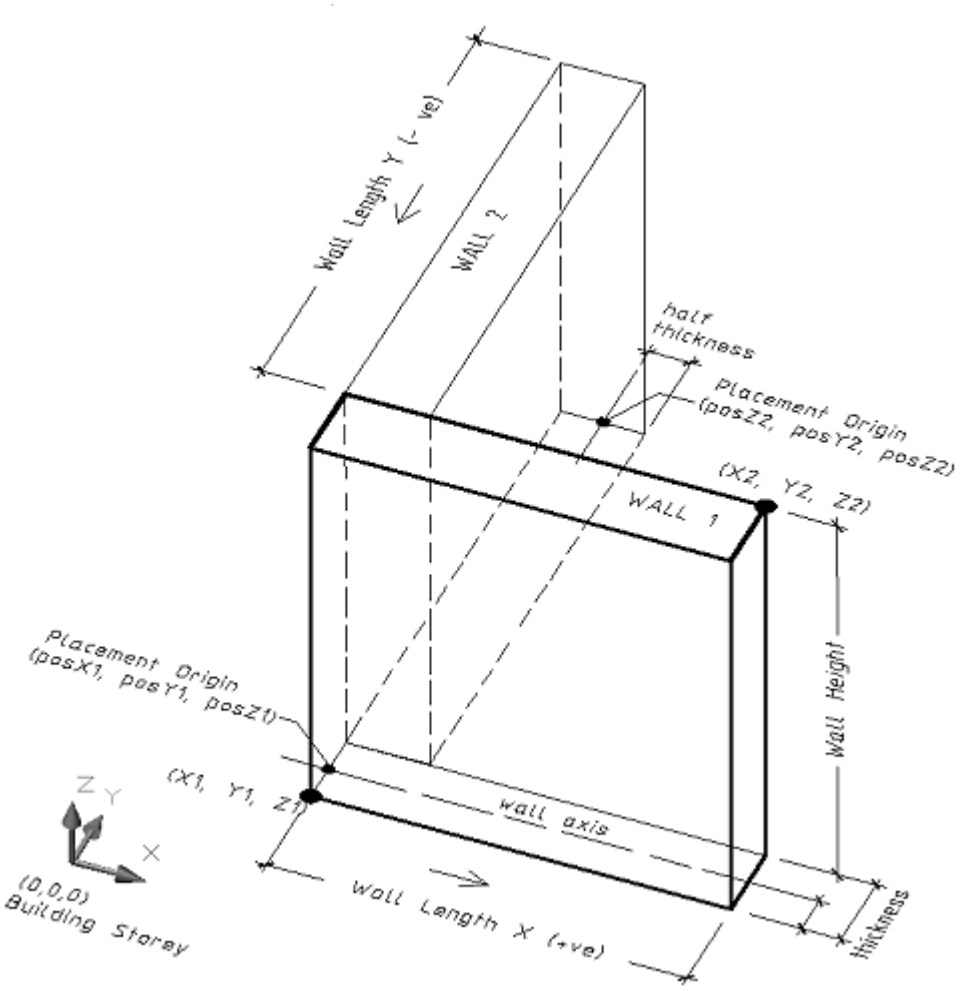


Figure 29: An *IfcWallStandardCase* representation

The wall body may be represented by one of three possible types of solid model, as given by the *IfcShapeRepresentation* entity:

1. Clipping representation
2. Swept Solid representation
3. Boundary Representation (BREP)

Clipping is the geometric representation as the result of a Boolean subtraction on the CSG solid models. Wall elements are often clipped by the roof entity or other clipping planes.

Swept Solid is the geometric representation of solid models created with profile sweeping using either linear extrusion or revolution techniques.

BREP describes a solid that is defined by its boundaries rather than by profiles and extrusions. In the IFC model, BREP geometry is the fallback position for any geometry than cannot reasonably be represented using parametric solids.

The wall representation type exchanged depends on the modelling methods used as well as on the IFC implementation in the BIM authoring application (Table 6). The following observations have been made based on the implementation of the Revit Building:

1. Walls having no "constraints" (i.e. detached from the floor and the ceiling) are always exchanged as *IfcWallStandardCase* regardless of whether or not they contain any doors, windows or wall openings. The wall body in this case is always represented as *Swept Solid* type.

2. When a wall has either a bottom or top constraint, there are two possibilities, as follows:

a) If it contains a door or a wall opening, it would be exchanged as *IfcWall* with *BREP* geometry.

b) If it is a solid wall without any openings, it would be exchanged as *IfcWallStandardCase* and with *Clipping* representation type.

Case	Top or bottom constraints	Openings Type	Wall Entity Type	Wall Body Type
1	No	Any opening	*IfcWallStandardCase*	*Swept Solid*
2	Yes	No openings	*IfcWallStandardCase*	*Clipping*
3	Yes	Door, window or wall opening	*IfcWall*	*BREP*

Table 6: Wall representation type exchanged

As shown in Table 6, there are three cases of wall representation. The types of wall representation and the wall height can be obtained by navigating into the STEP file data tree as follows:

Case 1: *IfcWallStandardCase*. The wall body is represented by SweptSolid type.

IfcWallStandardCase
    IfcProductDefinitionShape
        IfcShapeRepresentation (…, '**Body**', '**SweptSolid**', …)
            IfcExtrudedAreaSolid (***wall _ height***)

Where, ***wall _ height*** represents the height of the wall.

Case 2: *IfcWallStandardCase*. The wall body is represented by Clipping type.

IfcWallStandardCase
IfcProductDefinitionShape
IfcShapeRepresentation (…,'**Body**', '**Clipping**', …)
IfcGeometricRepresentationContext
IfcBooleanClippingResult
IfcExtrudedAreaSolid ($\boldsymbol{wall\_height}$)

Where, $\boldsymbol{wall\_height}$ represents the height of the wall.

Case 3: *IfcWall*. The wall body is represented by BREP geometry.

Two approaches can be used to determine the height of the wall for this case.

Approach 1:

Although the mapping of BREP geometry is not currently supported by ifcSTEP Parser, a relatively simple approach can be used to obtain the wall height. This involves navigating the STEP file hierarchy down to the *IfcPolyLoop* of the *IfcFacetedBrep* entity, and determining the maximum z coordinates of all the *IfcCartesianPoint* entities, as follows:

IfcWall
IfcProductDefinitionShape
IfcShapeRepresentation (…,'**Body**', '**Brep**',…)
IfcFacetedBrep
IfcClosedShell
IfcFaceOuterBound
IfcPolyLoop
IfcCartesianPoint (x1, y1, **z1**),
IfcCartesianPoint (x2, y2, **z2**),
IfcCartesianPoint (x3, y3, **z3**), etc.

Where, the maximum of **z1**, **z2**, **z3**, etc. is the height of the wall.

Approach 2:

Alternatively, a simpler approach can be used that is based on the fact that walls have a constant height within the same *IfcBuildingStorey*, which is the current limitation of ifcSTEP Parser.

At least one wall in the model must be represented by *IfcWallStandardCase.* Once this is found in the STEP file, the height of the wall can be obtained as per method given in Case 1 or Case 2 above. This is then taken to be the standard wall height and applied to all the walls on the same building storey.

This approach cannot be used where all walls in the model are represented by *ifcWalls*.

5.2.2 Dimensions of Walls

For Case 1 and Case 2 of the wall representation, the height of the wall can be obtained from the depth attribute of the *IfcExtrudedAreaSolid* entity, as described in the preceding section of this report.

For Case 3 of wall representation, two techniques for determining the wall height have also been suggested.

The length of the wall for both *IfcWallStandardCase* and *IfcWall* can be obtained by navigating into the wall axis shape representation entity in the STEP file data tree, as follows:

IfcWall or IfcWallStandardCase
  IfcProductDefinitionShape
    IfcShapeRepresentation (…,**'Axis'**, …)
      IfcGeometricRepresentationContext
      IfcPolyline
        IfcCartesianPoint
        IfcCartesianPoint (***wall _ length*** , 0.0)

Where, ***wall _ length*** represents the length on the wall centrelines, which has been adopted as the reference axis. Based on this convention, the walls can be positioned simply by using the given length and the thickness regardless of whether they have mitred or butt joints at corners (Figure 30).

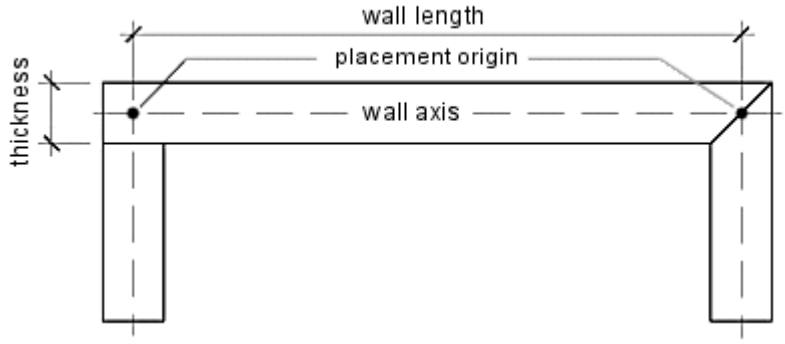


Figure 30: Wall length and placement origin on the wall axis

The wall thickness can be obtained directly from the *IfcMaterialLayer* in the STEP file.

5.2.3 Final Placement of Walls

The placement origin or the positional coordinates of the wall exchanged in the STEP file is assumed to be on the wall centreline, which is the adopted convention. This assumption may not always be true, as certain BIM authoring applications may have the default placement origin set on the outer or inner face of the wall rather than the centreline.

ifcSTEP Parser may need to be able to detect if the placement origin deviates from the centreline of the wall by identifying this property in the STEP file. In the case where they are not on the centreline, the correct length of the wall and the placement coordinates can be determined accordingly. This is particularly applicable when the walls have mitred joints in which case the length obtained from the STEP file could either be that of the outer or inner face.

The placement of walls depends on the positional coordinates as well as the directions, which can be obtained by navigating the STEP file data tree as follows:

IfcOpeningElement
  IfcLocalPlacement
    IfcLocalPlacement
      IfcAxis2Placement3D
        IfcCartesianPoint (pos_x , pos_y , pos_z)
        IfcDirection (0, 0, 1)
        IfcDirection (x_dir , y_dir , z_dir)

Where

pos_x , pos_y , pos_z are the positional coordinates of the wall expressed in the coordinate system of the building storey (*IfcBuildingStorey)*.

x_dir , y_dir , z_dir are wall directions in X, Y, Z axis respectively, which can be 0, +1, or -1.

The vector length of the wall can then be calculated by multiplying the length quantity by its direction, as follows:

$$(length\_x)_{wall} = (wall\_length) \times (x\_dir)$$

$$(length\_y)_{wall} = (wall\_length) \times (y\_dir)$$

As the walls are orthogonal with respect to each other and they can only be parallel to either X or Y axis as limited by the scope of this project, only one of the above vector components can represent the actual vector length of the wall. The other one would be zero.

For example, when $length\_x$ = 0 then the wall is in the Y-axis direction. Otherwise, it is in the X-axis direction. A positive value indicates that it is in the positive X or Y axis direction, and a negative value indicates that it is in the negative X or Y axis direction.

5.3 Mapping IFC Door and Window Openings

5.3.1 Door and Window Openings

Door and window openings are exchanged in STEP as *IfcOpeningElement*. The relationships between the openings and the voids in the wall or the openings and the filling elements such as a door or a window are described by *IfcRelVoidsElement* or *IfcRelFillsElement*, respectively.

Given an *IfcOpeningElement*, the associated wall can be determined from the *IfcRelVoidsElement*. In the STEP file, the last two attribute items in the *IfcRelVoidsElement* refer to the element IDs for the wall and the wall opening, respectively. Similarly, the door or window associated with the *IfcOpeningElement* can be determined from the *IfcRelFillsElement*, if it exists.

5.3.2 Dimensions of Door and Window Openings

The width, height, sill height and direction of the door and window openings (*IfcOpeningElement*) can be determined from the STEP data tree as follows:

IfcOpeningElement
- *IfcProductDefinitionShape*
  - *IfcShapeRepresentation*
    - *IfcExtrudedAreaSolid*
      - *IfcRectangleProfileDef* (…, …, …, ***Height***, ***Width***)
        - *IfcAxis2Placement2D*
          - *IfcCartesianPoint* (***z***1, ***w***1)
          - *IfcDirection* (x, y)
        - *IfcAxis2Placement3D*
          - *IfcCartesianPoint* (x,y,z)
          - *IfcDirection* (x, ***Opening-dir***, z)
          - *IfcDirection* (x,y,z)

Where

Height and ***Width*** are the height and width of the opening, respectively.

z1 and ***w1*** are offset distances to the centre of the opening, i.e.

$z1 = \left(\frac{Height}{2} \right)$ and $w1 = \left(\frac{Width}{2} \right)$, which can be used to verify the correct order for the ***Height*** and ***Width*** in the *IfcRectangleProfileDef* entity above.

Opening-dir is the swing direction of the door or the placement direction of a window, i.e. inward or outward in the case of doors, and inside or outside face in the case of windows.

5.3.3 Local Placement of Door and Window Openings

The local placement of door and window openings is exchanged as the horizontal and vertical offset distances from the placement origin of the wall to the edge of the opening. This can be obtained from the STEP file, as follows:

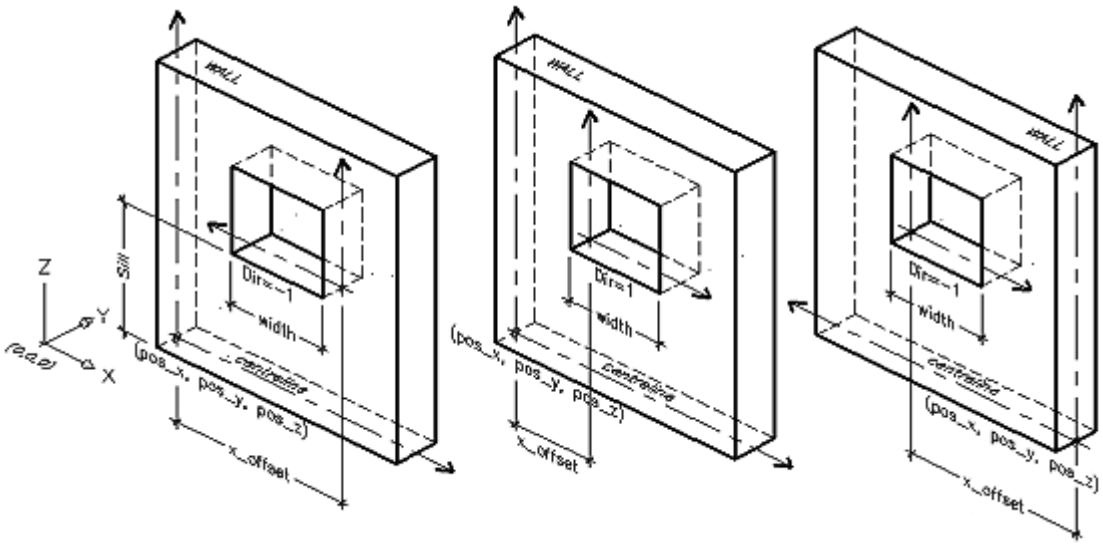
IfcOpeningElement
IfcLocalPlacement
IfcAxis2Placement3D
IfcCartesianPoint (x_offset , y_offset , z_offset *)*

Where,

x_offset is the horizontal offset distance (in X-axis direction) from the placement origin of the wall to either the near or far edge of the opening depending on the direction of the opening as determined in Section 5.3.2 above (refer also to Figure 31 a, b and c).

y_offset is the horizontal offset distance (in Y-axis direction) from the centreline of the wall to the face of the wall, i.e. half the thickness of the wall, which is not relevant for openings through the wall. However, this may become relevant for wall recesses, which are not considered in this project.

z_offset is the vertical offset distance (in the Z-axis direction) from the placement origin of the wall (or the base of the wall) to the sill of the opening (Figure 31 a).



(a) Wall +ve, Opening –ve **(b) Wall +ve, Opening +ve** **(c) Wall -ve, Opening +ve**

Figure 31: Placement of Door and Window Openings

Figure 31 (a) shows the wall in the positive X-axis direction and the opening is in the opposite direction. The x_offset of the opening is the distance from the placement origin of the wall to the far edge of the opening. The z_offset is the distance from the base of the wall to the sill of the opening.

Figure 31 (b) shows the wall and the opening are both in the positive X-axis direction and the x_offset distance is taken to the near edge of the opening. In this case, the placement origin of the opening is at the edge of the opening closest to the placement origin of the wall. This is the preferred situation and has been adopted as the convention for the placement of door and window openings. In the case where the placement origin of the opening is positioned at the far edge of the opening relative to that of the wall (such as shown in

Figure 31 (a and c), the preferred placement origin of the opening can be determined by subtracting the width of the opening from the x_offset .

Figure 31 (c) shows the wall in the negative X-axis direction and the opening is in the positive X-axis direction. The x_offset distance is taken to the far edge of the opening. Therefore, the preferred placement origin of the opening would need to be calculated by subtracting the width of the opening.

5.3.4 Direction of Doors and Windows Relative to the Parent Wall

Doors swing direction can either be inward or outward. Similarly, windows placement can either be on one side or the other relative to the parent wall. As previously described, the placement of doors and windows is given by the horizontal and vertical offset distances relative to the local placement of the parent wall.

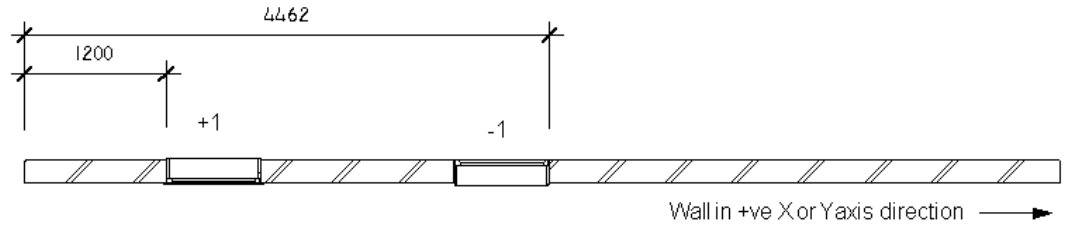


Figure 32: Windows configuration on a wall in positive X-axis direction

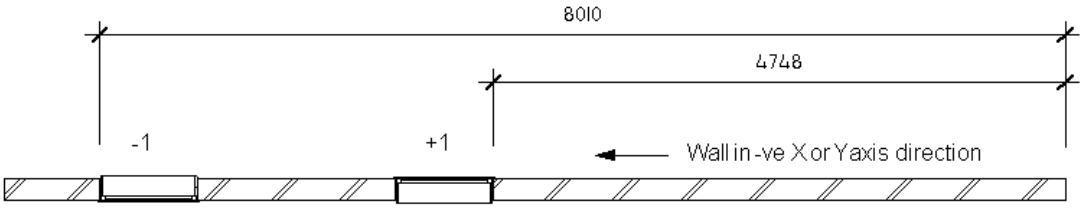


Figure 33: Windows configuration on a wall in negative X-axis direction

As FDS is generally interested only with the actual door opening, the door swing direction is of no particular relevance. Therefore, both windows and doors have exactly the same cases of placement directions relative to the parent wall, i.e. it's either positive or negative relative to the parent wall.

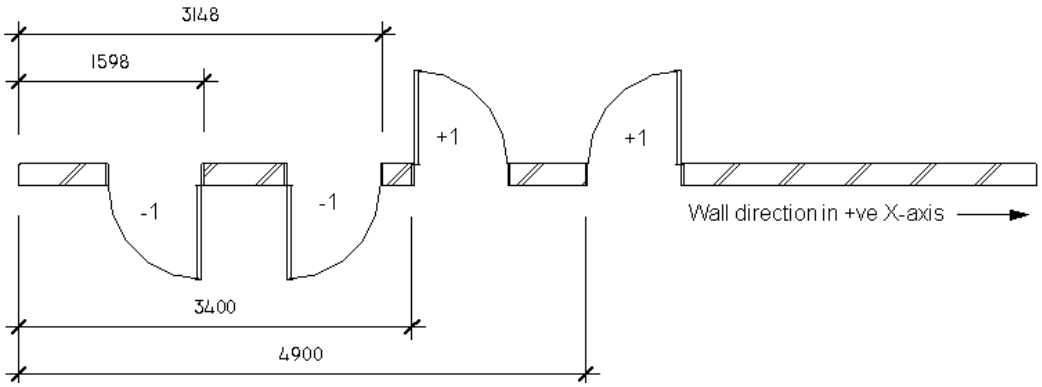


Figure 34: Doors configuration on a wall in positive X-axis direction

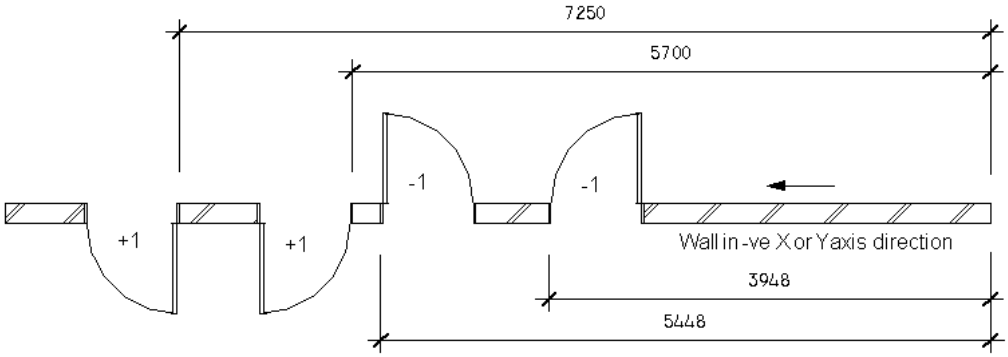


Figure 35: Doors configuration on a wall in positive X-axis direction

The horizontal offset distance is taken to either the near or far edge of the opening depending on the direction of the opening relative to that of the parent wall (refer to examples shown in Figure 32, Figure 33, Figure 34, and Figure 35).

This relative direction between the wall and the openings can be summarised as follows:

1. The wall opening is in the same direction as the parent wall, denoted with +1. The offset distance is to the near edge of the opening relative to the local placement of the parent wall.

2. The wall opening is in the opposite direction to the parent wall, denoted with -1. The offset distance is to the far edge of the opening relative to the local placement of the parent wall.

Therefore, it is convenient to allow the wall openings to inherit the direction of the parent wall. This is achieved by multiplying the width of the opening by the directions of the parent wall, which results in two vector components, as follows:

$$\boldsymbol{width\_x} = \left(\boldsymbol{Width} \times \boldsymbol{x\_dir}\right)$$

$$\boldsymbol{width\_y} = \left(\boldsymbol{Width} \times \boldsymbol{y\_dir}\right)$$

Where $\boldsymbol{Width}$ is the width of the opening obtained from the STEP file as described in Section 5.3.2.

As the walls are orthogonal with respect to each other and restricted to those parallel to either X or Y axis, only one of these vector components can represent the actual vector length (or width) of the opening. The other one would be zero.

For example, when $\boldsymbol{width\_x}$ = 0 then the door or window opening is parallel to the Y axis. Otherwise, it is in the X axis direction. A positive value indicates that it is in the positive X or Y axis direction, and a negative value indicates that it is in the negative X or Y axis direction.

5.3.5 Final Placement of Door and Window Openings

In accordance with the adopted convention, the placement origin or the positional coordinates of the door and window openings must be on the wall centreline or the reference axis (Figure 30).

For the two cases of the relative direction between the wall and the openings as described in the preceding section, the final placement origin of the door and window openings can be determined as follows:

Case 1: Door and window opening direction = +1

1. For walls parallel to the X axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall} + (x\_offset \times x\_dir)$$

$$(pos\_y)_{opening} = (pos\_y)_{wall}$$

$$(pos\_z)_{opening} = (pos\_z)_{wall} + z\_offset$$

2. For walls parallel to the Y axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall}$$

$$(pos\_y)_{opening} = (pos\_y)_{wall} + (x\_offset \times y\_dir)$$

$$(pos\_z)_{opening} = (pos\_z)_{wall} + z\_offset$$

Case 2: Door and window opening direction = -1

1. For walls parallel to the X axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall} + (x\_offset \times x\_dir) - (width\_x)$$

$$(pos\_y)_{opening} = (pos\_y)_{wall}$$

$$(pos\_z)_{opening} = (pos\_z)_{wall} + z\_offset$$

2. For walls parallel to the Y axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall}$$

$$(pos\_y)_{opening} = (pos\_y)_{wall} + (x\_offset \times y\_dir) - (width\_y)$$

$$(pos\_z)_{opening} = (pos\_z)_{wall} + z\_offset$$

5.4 Mapping IFC Wall Openings

5.4.1 Dimensions of Wall Openings

In the context of this report, a wall opening is a void or hole through the wall without a door or a window. The wall opening objects must be treated differently as they differ in details to doors and windows openings and are exchanged differently in the STEP file.

The height and width, and the local placement of the wall opening can be obtained from the STEP file, as follows:

IfcOpeningElement
IfcProductDefinitionShape
IfcShapeRepresentation
IfcExtrudedAreaSolid
IfcRectangleProfileDef (…, …, …, ***Height*** , ***Width***)
IfcAxis2Placement2D
IfcAxis2Placement3D
IfcCartesianPoint (x_offset , y_offset , z_offset)

Where, ***Height*** and ***Width*** are the height and width of the opening, respectively.

IfcAxis2Placement3D gives the local placement coordinates of the wall opening at the centre of the opening, i.e.

x_offset = Horizontal offset distance from the placement origin of the wall to the centre of the opening in X axis direction.

y_offset = Horizontal offset distance in Y axis direction from the wall axis to the face of the wall, i.e. half the thickness of the wall.

z_offset = Vertical offset distance in Z axis direction from the placement origin of the wall (or base of the wall) to the centre of the opening.

As described in wall representation Case 3, when a wall opening is on a wall that is attached to the floor (i.e. with base constraints), the wall would be exchanged as *IfcWall* with BREP representation. In this case, the opening would be part of the BREP profile of the wall geometry and would not be exchanged as *IfcOpeningElement* entity. This is consistent with the observations obtained from test case models (Chapter 7.0 and Appendix 2).

It has been observed that in order for the wall opening to be correctly exchanged as *IfcOpeningElement*, the base constraints of the wall must be removed in the source Revit Building model.

5.4.2 Final Placement of Wall Openings

Unlike the door and window openings, the local placement of the wall openings is at the centre of the opening instead of at the edge (Figure 36).

In accordance with the adopted convention, the final placement of the opening needs to be shifted from the centre to the edge closest to the placement origin of the wall. This can be achieved by subtracting half of the width of the opening from the x_offset, and subtracting half of the height of the opening from the z_offset, as follows:

1. For walls parallel to the X axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall} + (x\_offset \times x\_dir) - \left(\frac{width\_x}{2}\right)$$

$$(pos\_y)_{opening} = (pos\_y)_{wall}$$

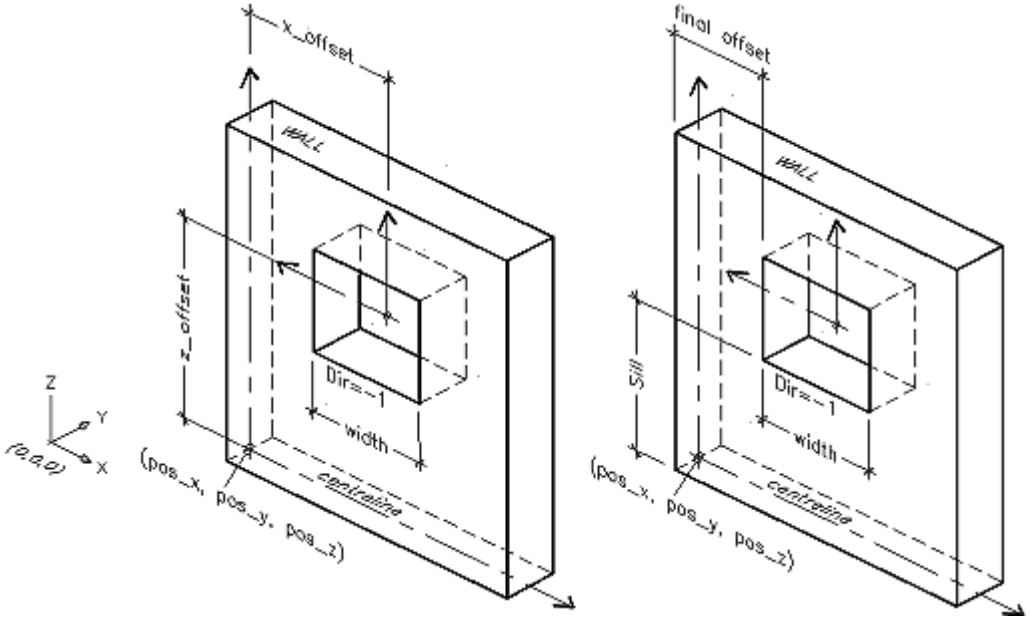
$$(pos\_z)_{opening} = z\_offset - \left(\frac{Height}{2}\right)$$

2. For walls parallel to the Y axis:

$$(pos\_x)_{opening} = (pos\_x)_{wall}$$

$$(pos\_y)_{opening} = (pos\_y)_{wall} + (x\_offset \times y\_dir) - \left(\frac{width\_y}{2}\right)$$

$$(pos\_z)_{opening} = z\_offset - \left(\frac{Height}{2}\right)$$



(a) Revit Building model IFC representation **(b) Final opening placement**

Figure 36: Placement of wall openings

5.5 Mapping IFC Furniture Placement

5.5.1 Final Placement of Furniture Entities

For FDS simulations, it would be useful to be able to place a box in the middle of a room to represent a piece of furniture or a group of fuel packages. This can be achieved by incorporating a piece of furniture item in the source model, which would be exchanged as *IfcFurnishingElement*.

Given the current limitations of ifcSTEP Parser to resolve Brep geometry, complex shapes such as common furniture items cannot yet be mapped. However, the position coordinates of the furniture item can be exchanged relatively easily.

The position coordinates of the furniture item can be obtained from the STEP file as follows:

IfcFurnishingElement
    IfcLocalPlacement
        IfcAxis2Placement3D
            IfcCartesianPoint (pos_x , pos_y , pos_z)

pos_x , pos_y , pos_z are the position coordinates of the furniture item. For the purposes of mapping to FDS, this is assumed to be at the centre of the base of a representative furniture item (Figure 37).

This assumption may not always be accurate as the position coordinates of the BIM furniture objects may be at different locations. However, for small furniture items located in the centre of the room and not in close proximity to any walls, the slight offset of the position coordinates is considered to be insignificant in terms of fire simulation.

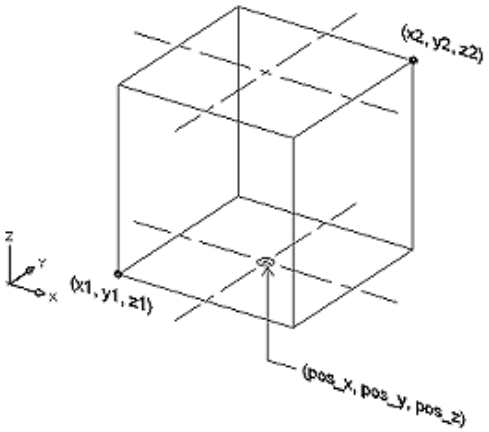


Figure 37: Placement of a furniture item

6.0 IfcSTEP-FDS application

6.1 Initial Development

ifcSTEP-FDS is a software application developed as part of this research project as an external interface to ifcSTEP Parser. It is intended for parsing the XML output of ifcSTEP Parser and generating the input data required for the FDS simulations.

IfcSTEP-FDS was initially developed as a stand-alone desktop application for the Microsoft Windows operating system using the Microsoft .NET programming framework. The XML parsing functionalities were achieved using the ADO.Net DataSets technology provided by the Microsoft .NET 2005 framework.

The graphical user interface (GUI) was used to display various input data including the selected XML elements in a bound DataGrid object, and also provided the option to generate the FDS input file. For user's convenience, FDS and SmokeView applications could be executed directly from the application. Refer to Figure 38 for the screen shot of the GUI of this initial version of ifcSTEP-FDS.

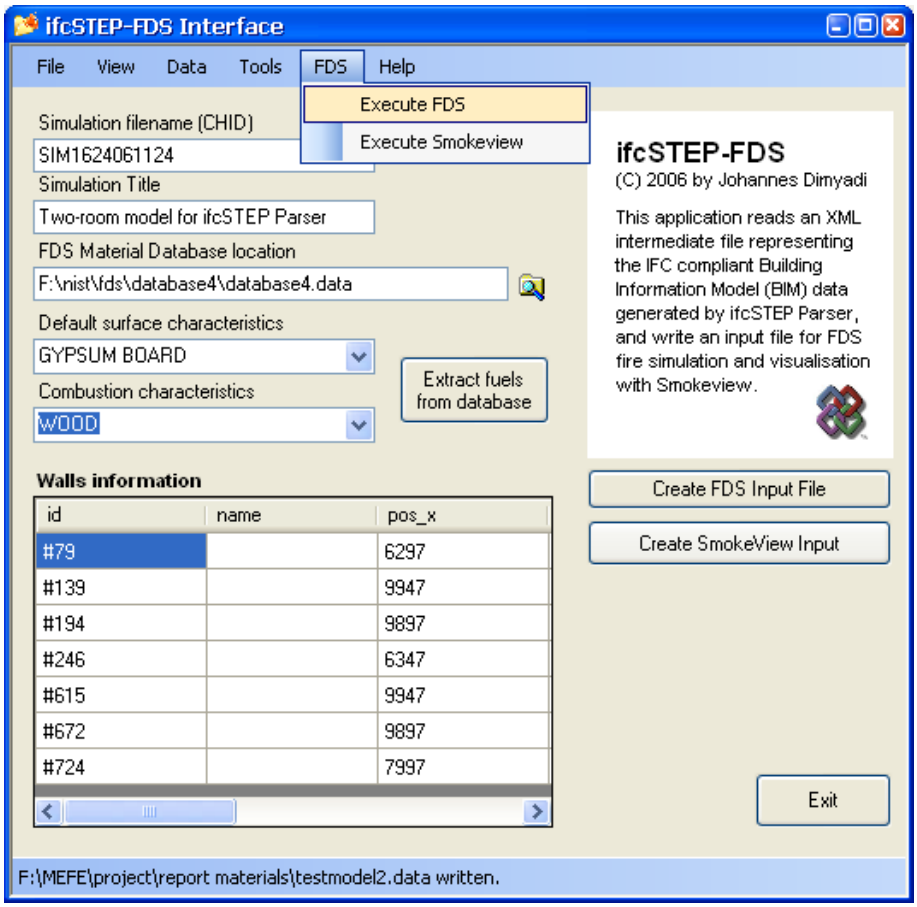


Figure 38: GUI of the initial version of ifcSTEP-FDS

The development of this initial stand-alone version of ifcSTEP-FDS was abandoned in place of a new web-based version, which was considered to be a better option as it would provide a centralised installation that can be accessed and maintained more efficiently and effectively.

One obvious advantage of a web application is that it can be accessed by anyone using a web browser anywhere provided that there is an internet connection. It is also practical to be able to update and maintain the application in one central location without the need for the distribution and the installation of software components on the users' computers.

6.2 Web Application Development

The feasibility of developing ifcSTEP-FDS as a web application depends on a number of factors at the outset, as follows:

1. The selection of a reliable web server is required to host the application.
2. The choice of the programming language used to develop the application, which must be supported by the web server.
3. The need to upload an XML document from the user's local computer for parsing.
4. The application must be able to process and return a suitable output to the user on the fly.

In the context of web applications development, there is a host of options and programming languages available. However, a set of criteria has been established for the purposes of choosing a suitable programming language for the development, as follows:

1. The language must be platform independent and widely used for web applications development.
2. The ability to upload a file from the user's local computer and to parse an XML document.
3. Capable of server-side processing and embeddable into HTML.
4. Support for some graphics programming functions.
5. Well documented with a good repository of resources available.

Basic commercial web hosting services do not often support web development languages other than PHP, which is a widely-used, open-source, freely available, and well documented server-side scripting language that can be embedded into HTML. PHP has a very similar syntax to C and has a comprehensive set of online documentation, which makes it relatively easy to use [Brown, 2002]. It is also native to the Apache web server, which is a popular server among web hosting companies due to its good security track record. PHP is an efficient language having a relatively short code-path, which would speed up server-side processing hence returning results to the web browser faster. For these reasons, PHP has been adopted as the development language for ifcSTEP-FDS.

The main page of the ifcSTEP-FDS consists of only approximately 300 lines of HTML with embedded PHP codes. Each function of the application is handled by a separate module that is server-side included on demand. The size of these modules ranges between 200 and 400 lines of mainly PHP code embedded within the HTML.

IfcSTEP-FDS uses several external libraries which are all available as part of the PHP implementation on the Apache server, such as the Expat library to parse XML documents and the GD library to dynamically generate graphics.

6.3 User Interface

The user interface of ifcSTEP-FDS is the web application interface. This has been designed as a single web page occupying 750 pixels screen wide that would fit the width of a computer screen of 800 x 600 pixels resolution size without the need for horizontal scrolling. The page header has a static banner displaying the application title and a description with some representative graphical images. The page footer contains other information including the application version number and the last modified date. The area between the header and the footer is the main body of the application web page, which is used both as the input screen as well as to display the application output.

Initially and on subsequent web browser refresh, the main body area would contain HTML form objects, which include input text boxes and command buttons and a set of notes describing the functionalities of the application (Figure 39). The application output displayed in this area depends on which one of the four application functions is selected by the user, i.e. Parse, Brief, Geometry, and FDS. Refer Section 6.9 for details of the output for each function.

At the bottom of the main body just above the footer area, a line of hyperlinks has been incorporated to allow users to access the URLs of various relevant organisations. A number of graphics symbols are also displayed in this area to acknowledge that PHP has been used as the development language, and also to certify that the web page mark-ups have been validated against W3C specifications for web interoperability.

The control of styles and presentation of the application web page has been achieved by the use of cascading style-sheet (css), which has also been validated against the W3C specification. The contents of the css include parameters to control text position, styles and sizes, colour scheme, as well as other HTML elements and their properties.

6.4 Input Requirements

The only input required by ifcSTEP-FDS is an XML document generated by ifcSTEP Parser, which contains only relevant fire-engineering subset of IFC entities of the model.

Using a standard HTML Input File form object provided, the user can browse the local computer file storage system to select and upload the input XML document for processing (Figure 39).

Figure 39: Screenshot of ifcSTEP-FDS web interface

The user may optionally specify the material database file installed with the FDS application on the local computer. If specified, the name of the file would be used in the application output. The types of gas and solid phase materials defined in the database would also be retrieved and used to populate the combo boxes displayed above the output text for user's convenience and useful reference (Figure 53).

6.5 Application Flow Diagram

Like any computer application, the ifcSTEP-FDS web application has an input, a set of processes and an output. This is illustrated in a general flow diagram shown in Figure 40.

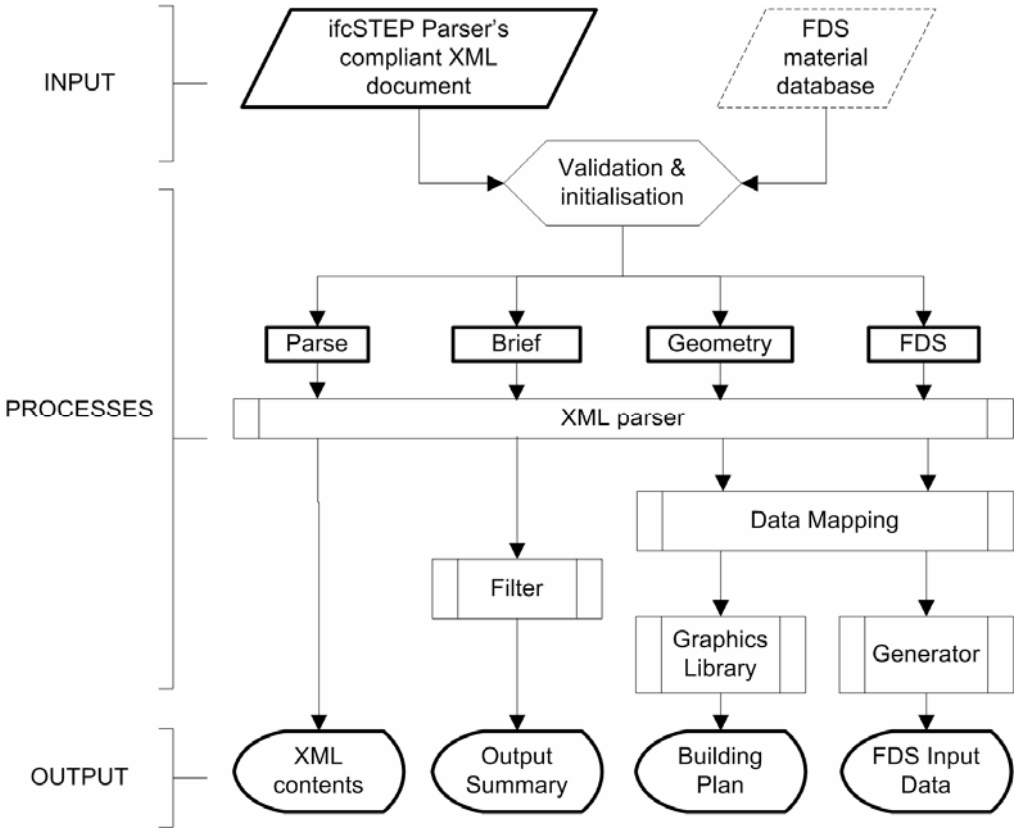


Figure 40: ifcSTEP-FDS Application Flow Diagram

Once the XML input document is selected or specified by the user, a copy of the document is uploaded to the temporary area of the web server for parsing by the XML parser of the application.

There are three levels of processes involved after the input document has been parsed depending on which function of the application is selected, as follows:

1. The contents and attributes of the XML document is either displayed in its entirety or filtered so that only relevant elements are displayed, i.e. when the “Parser” function is selected.

2. The selected contents and attributes of the XML documents are submitted to a data mapping process, i.e. when “Brief” is selected.

3. The mapping results are used to either produce a graphical display of the building plan or output as the FDS input data, i.e. when either the “Geometry” or “FDS” is selected.

6.6 XML document Parsing

IfcSTEP-FDS utilises a set of extended functions provided by PHP to enable XML parsing using an implementation of the Expat Library. A snippet of the parser setup procedure code is given in the Appendix A6.1. The procedure creates the parser object, sets up the element tag handler, opens the XML document and feeds the contents to the parser.

In navigating through the XML document, the parser identifies various elements and picks up the specified attributes and assigned them to variables for further processing. This can be seen in a code snippet given in Appendix A6.2

The attributes retrieval function also validates the XML document for compliance with the ifcSTEP Parser's document output specification. This is currently achieved by identifying an element named "STEPPARSER-DATA-CONSTRUCTION" in the header of the document. In the absence of this element, the XML document specified would be considered non-compliant and rejected for further processing.

6.7 Graphical Output

For the graphical output of the building plan, ifcSTEP-FDS uses PHP functions provided by the GD Library implementation. The code snippet for the graphics generation procedure is given in Appendix A6.3.

GD library functions are employed to generate a graphical object containing the image of the building plan in a predefined format, which can either be output to a file or rendered directly onto the web page by using the appropriate output header specifications in PHP.

The GIF graphics format has been used in this case as it is reasonably compact and optimised for the colour scheme and the resolution size adopted.

6.8 Mapping the XML input data

6.8.1 Mapping of XML document header information

At the top of the XML input document are some header information specific to the ifcSTEP Parser. They identify the type of the document, version number, the name of the author and the associated organisation, the creation date and time of the document, the project ID and title, as well as the unit of measurement (Figure 41).

As shown in Figure 41, the unique id numbers assigned to the "project" and "units" entities have been mapped exactly from those used by the corresponding IFC entities in the original STEP file.

```
<STEPParser-data-construction version="0.2" author="Michael Spearpoint" organisation="University of Canterbury"
        created="14:03:38 on Friday 15 September 2006">
<project id="#8716" name="TWOROOM2J-REVIT9">
        <units id="#8748" length="MILLIMETRE" />
</project>
```

Figure 41: XML document header information

The header information is parsed by ifcSTEP-FDS and transferred to the FDS input data. The data mapping process is illustrated in Figure 42.

The diagram shows the mapping of individual attributes of a particular element between the source XML document and the target FDS input data schemas. Any calculations required as part of the mapping are also shown where applicable.

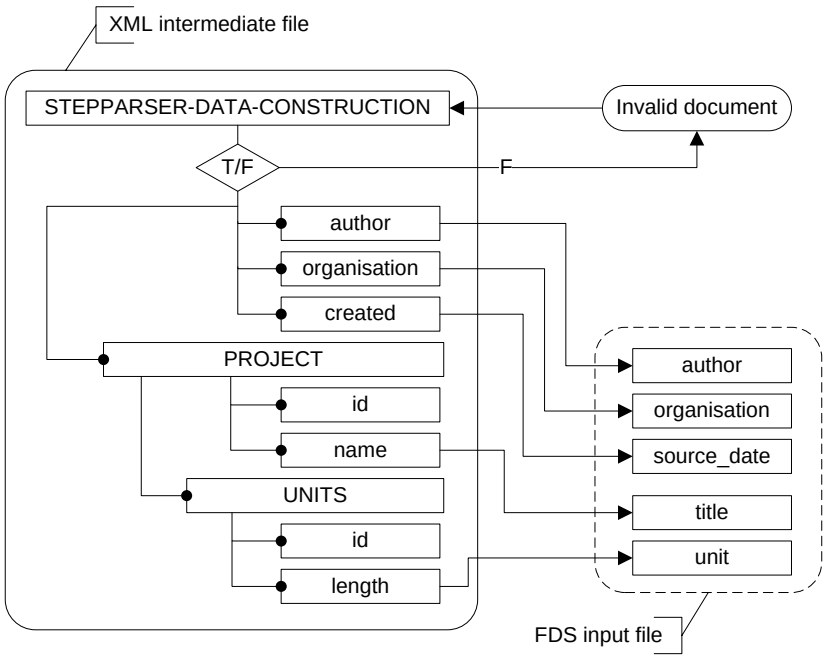


Figure 42: Header information mapping diagram

6.8.2 Mapping of Wall Entities

The wall geometry and its placement properties are given in the XML document as the attributes of the wall element (Figure 43).

```
<wall id="#79" name="" pos_x="6297" pos_y="9691" pos_z="150" length_x="3700" length_y="0" height="2400">
  <materials id="#88">
    <material id="#87" description="Default Wall" thickness="100" />
  </materials>
</wall>
<wall id="#139" name="" pos_x="9947" pos_y="9641" pos_z="150" length_x="0" length_y="-2400" height="2400">
  <materials id="#88">
    <material id="#87" description="Default Wall" thickness="100" />
  </materials>
</wall>
<wall id="#194" name="" pos_x="9897" pos_y="7291" pos_z="150" length_x="-3600" length_y="0" height="2400">
  <materials id="#88">
    <material id="#87" description="Default Wall" thickness="100" />
  </materials>
</wall>
```

Figure 43: Extract of an XML document showing attributes for the walls

The position coordinates given in the XML document refer to the wall centreline. To determine the sextuplet coordinates, i.e. x1, x2, y1, y2, z1, z2 required to define a wall as a solid obstruction block in FDS, the following mapping methodology has been used (also refer to Figure 44 and Figure 45):

1. The wall direction is given by either $\boldsymbol{length\_x}$ or $\boldsymbol{length\_y}$ in the XML input document (Figure 43). If $\boldsymbol{length\_x}$ is zero, then the wall is parallel to the Y-axis, otherwise it is in the X-axis direction.

2. Determine the amount of offset from the centreline to the face of the wall, i.e.

$$wall\_offset = \left(\frac{wall\_thickness}{2} \right)$$

3. For walls parallel to the X axis, the following calculations have been used to determine the required coordinates.

$$x1 = pos\_x$$
$$y1 = pos\_y - (wall\_offset)$$
$$z1 = pos\_z$$
$$x2 = pos\_x + (length\_x)$$
$$y2 = pos\_y + (wall\_offset)$$
$$z2 = pos\_z + (height)$$

4. For walls parallel to the Y axis, the following calculations have been used to determine the required coordinates.

$$x1 = pos\_x - (wall\_offset)$$
$$y1 = pos\_y$$
$$z1 = pos\_z$$
$$x2 = pos\_x + (wall\_offset)$$
$$y2 = pos\_y + (length\_y)$$
$$z2 = pos\_z + (height)$$

Figure 44 and Figure 45 show the mapping diagrams for walls parallel to X and Y axis, respectively.

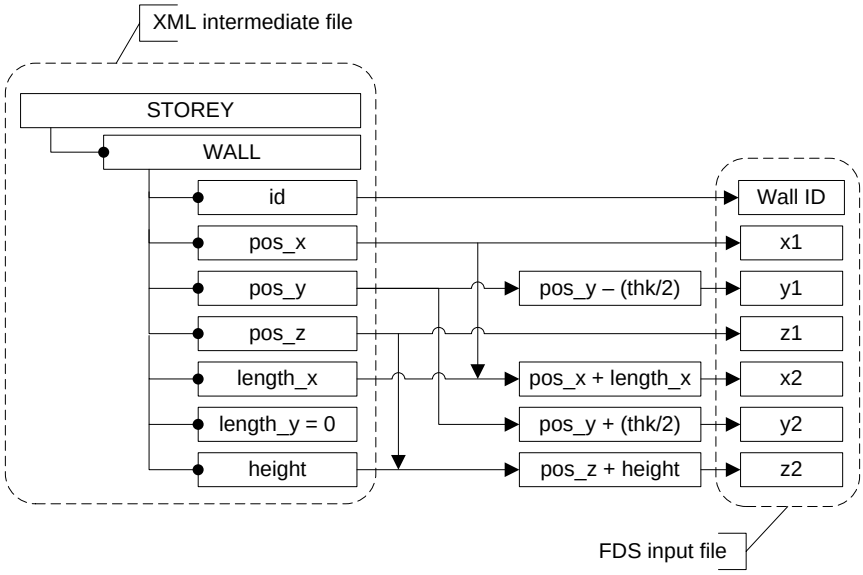


Figure 44: Mapping diagram for walls parallel to the X axis

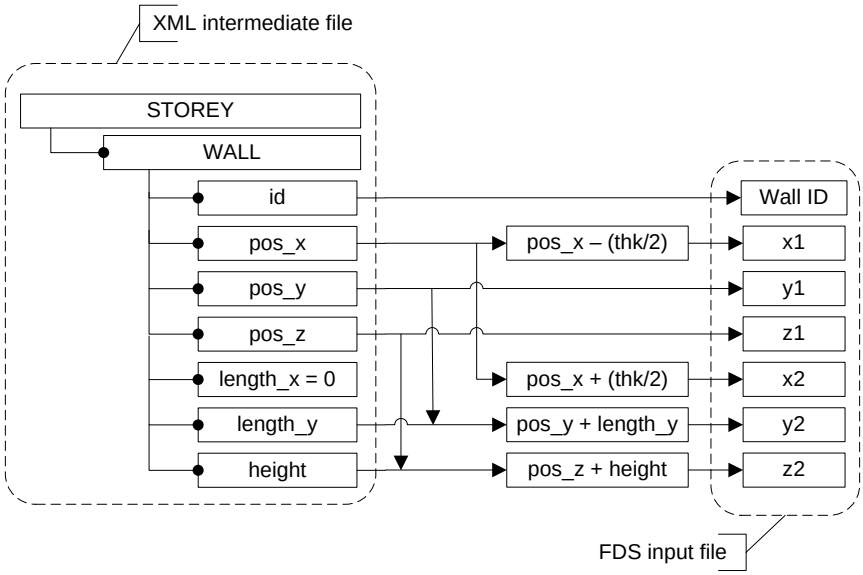


Figure 45: Mapping diagram for walls parallel to the Y axis

6.8.3 Mapping of Wall Openings

The placement properties and geometry of wall openings are given in the XML input document as the attributes of the door, window or opening elements under the parent wall element (Figure 46).

```
<wall id="#18" name="wall1234" length_x="0" length_y="14600" height="2550" pos_x="747" pos_y="5341" pos_z="0">
 <door id="#22" name="1" width_x="0" width_y="758" height="1980" soffit="2130" pos_x="747" pos_y="888" pos_z="0" />
 <door id="#29" name="2" width_x="0" width_y="758" height="1980" soffit="2130" pos_x="747" pos_y="288" pos_z="0" />
 <window id="#25" name="3" width_x="0" width_y="-810" height="1010" sill="900" soffit="1910" pos_x="647" pos_y="778"
pos_z="0" />
<materials id="#95">
   <material id="#94" description="Default Wall" thickness="100" />
 </materials>
</wall>
```

Figure 46: Extract of an XML document showing door and window openings

To determine the sextuplet coordinates, i.e. x1, x2, y1, y2, z1, z2 required to define a wall opening as a void in FDS, the following mapping methodology has been used in the application:

1. The opening direction is given by either $\boldsymbol{width\_x}$ or $\boldsymbol{width\_y}$ in the XML input document (Figure 46). If $\boldsymbol{width\_x}$ is zero, then the wall is parallel to the Y axis, otherwise it is in the X axis direction.

2. Determine the amount of offset from the centreline to the face of the wall, i.e.

   $$wall\_offset = \left(\frac{wall\_thickness}{2} \right) \times 1.5$$

   The arbitrary 1.5 factor in the equation represents 25% additional wall thickness on each side of the centreline to ensure that the hole would punch through the entire thickness of the obstruction block in FDS.

3. For openings in the X-axis direction, the following calculations have been used to determine the required coordinates.

   $$x1 = pos\_x$$
   $$y1 = pos\_y - \left(wall\_offset \right)$$
   $$z1 = pos\_z$$
   $$x2 = pos\_x + \left(width\_x \right)$$
   $$y2 = pos\_y + \left(wall\_offset \right)$$
   $$z2 = pos\_z + \left(height \right)$$

4. For walls in the Y-axis direction, the following calculations have been used to determine the required coordinates.

$$x1 = pos\_x - (wall\_offset)$$
$$y1 = pos\_y$$
$$z1 = pos\_z$$
$$x2 = pos\_x + (wall\_offset)$$
$$y2 = pos\_y + (width\_y)$$
$$z2 = pos\_z + (height)$$

Figure 47 and Figure 48 show the mapping diagrams for openings on walls parallel to X and Y axis, respectively.

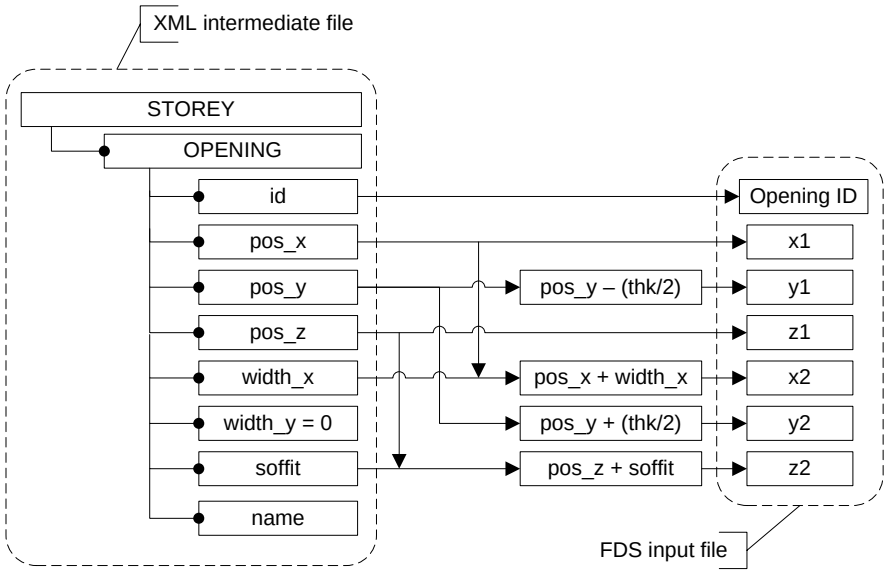


Figure 47: Mapping diagram for openings on walls parallel to the X axis

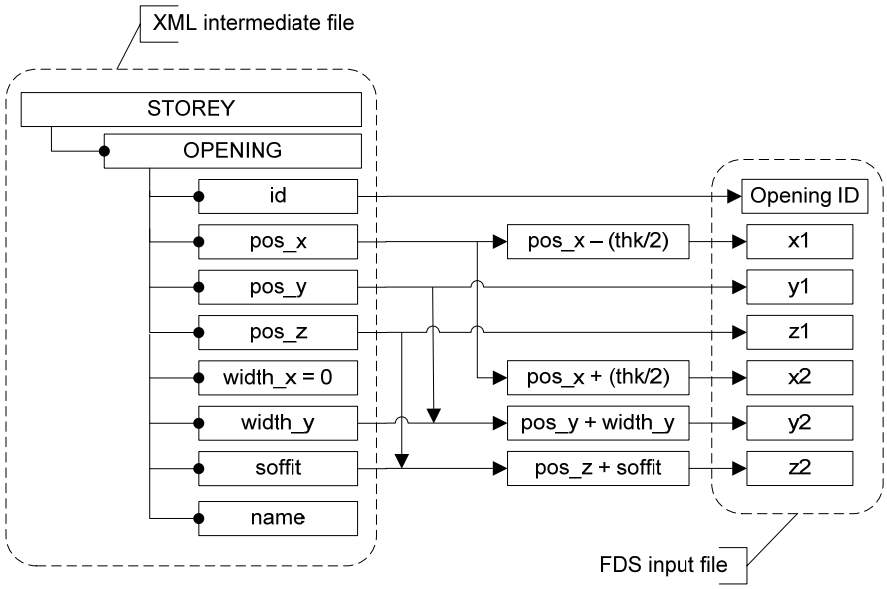


Figure 48: Mapping diagram for openings on walls parallel to the Y axis

6.8.4 Mapping of Furniture Items

Given the current limitation of ifcSTEP Parser to resolve complex shapes represented by BREP geometry, a relatively simple approach has been suggested for mapping the location of a furniture item to FDS. The actual shape of the furniture item is not mapped but is instead replaced with a representative box object of a predetermined size (refer Section 5.5).

The placement of the furniture items are given in the XML input document as the position coordinates attributes of the furniture element (Figure 49) as suggested in Section 5.5.

```
<furniture id="#1504" name=" SimpleBox:113595" pos_x="8356" pos_y="6279" pos_z="148" />
</storey>
```

Figure 49: Extract of an XML document showing a furniture element

The representative box is currently given a dimension of 150mm x 150mm and 300mm high. The default surface material applied to the box is that of upholstery as defined in the generated FDS input data. The top surface of the box is specified with a predefined ignition source using one of the prescribed design fires. The default parameters in the generated FDS input file can be edited using any text editors prior to the fire simulation.

To determine the sextuplet coordinates, i.e. x1, x2, y1, y2, z1, z2 required to define a box measuring 150 mm x 150 mm x 300 mm representing the furniture item in FDS, the following mapping methodology has been used:

$$x1 = pos\_x - (150 \times length\_factor)$$
$$y1 = pos\_y - (150 \times length\_factor)$$
$$z1 = pos\_z$$
$$x2 = pos\_x + (150 \times length\_factor)$$
$$y2 = pos\_y + (150 \times length\_factor)$$
$$z2 = pos\_z + (300 \times length\_factor)$$

Where $length\_factor$ is the unit conversion factor, i.e. if the unit of length measurement is in metre, then $length\_factor$ = 0.001

The mapping algorithm is shown in a code snippet given in the Appendix A6.6.

6.9 *Application Output Types*

There are four different types of output corresponding to the four key features currently implemented in ifcSTEP-FDS, namely XML Parsing, Summary Output, Graphical Output, and FDS Generated Input Data. These are described in the following sections of the report.

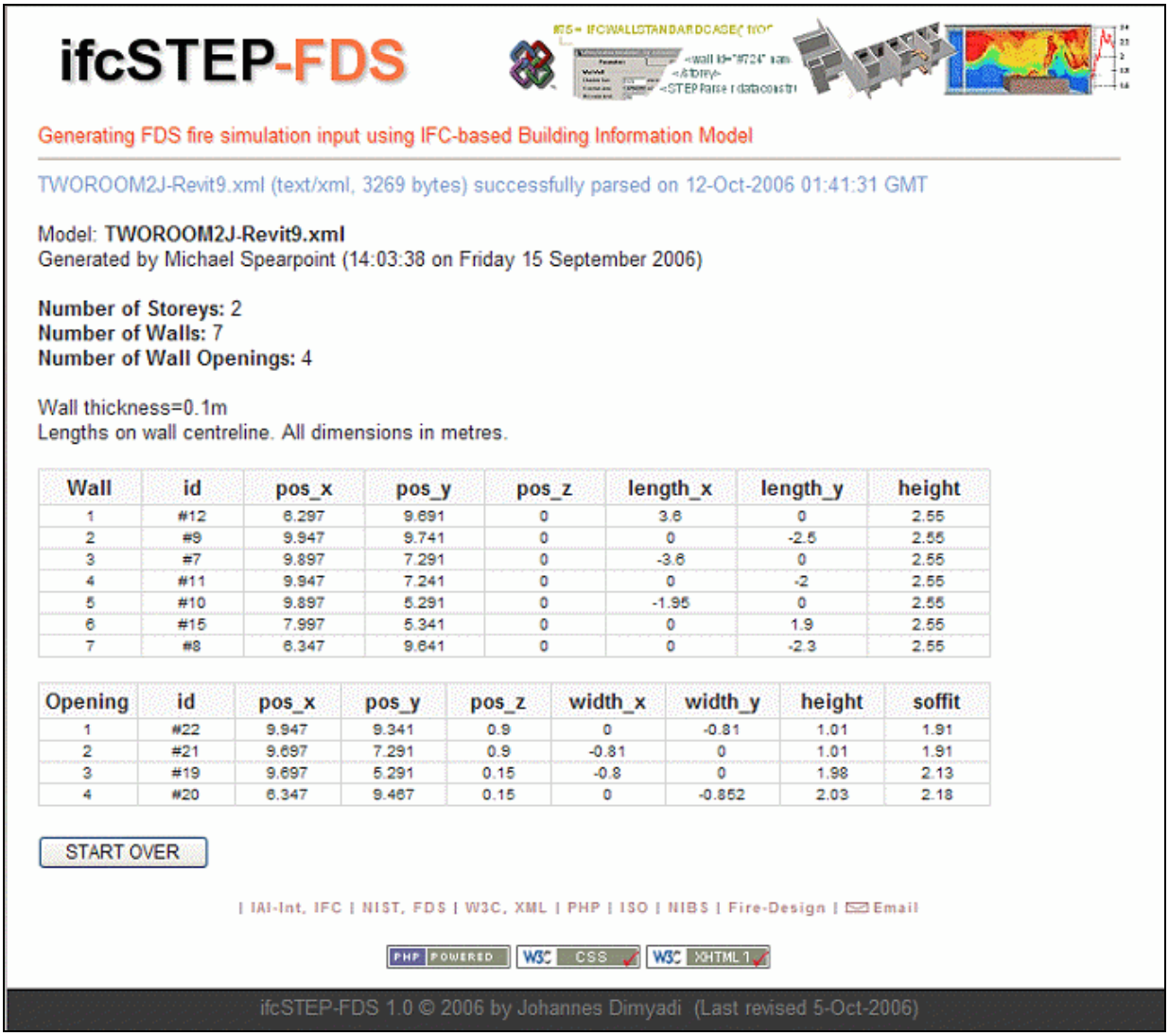
6.9.1 XML Document Parsing Full Output

Parses the ifcSTEP Parser's compliant XML document specified and displays on-screen the complete list of elements and attributes found in the document (Figure 50).

Figure 50: Screenshot of ifcSTEP-FDS Parse Output

6.9.2 XML Document Parsing Summary Output

Parses the ifcSTEP Parser's compliant XML document specified and displays on-screen a summary of walls and wall openings and associated attributes for mapping to the FDS input data (Figure 51).



Wall	id	pos_x	pos_y	pos_z	length_x	length_y	height
1	#12	6.297	9.691	0	3.6	0	2.55
2	#9	9.947	9.741	0	0	-2.5	2.55
3	#7	9.897	7.291	0	-3.6	0	2.55
4	#11	9.947	7.241	0	0	-2	2.55
5	#10	9.897	5.291	0	-1.95	0	2.55
6	#15	7.997	5.341	0	0	1.9	2.55
7	#8	6.347	9.641	0	0	-2.3	2.55

Opening	id	pos_x	pos_y	pos_z	width_x	width_y	height	soffit
1	#22	9.947	9.341	0.9	0	-0.81	1.01	1.91
2	#21	9.697	7.291	0.9	-0.81	0	1.01	1.91
3	#19	9.697	5.291	0.15	-0.8	0	1.98	2.13
4	#20	6.347	9.467	0.15	0	-0.852	2.03	2.18

Figure 51: Screenshot of ifcSTEP-FDS Summary Output

6.9.3 Building Plan Geometry Output

The feature parses the ifcSTEP Parser's compliant XML document specified and generates the building plan geometry for display on-screen along with the mapping results for the walls and wall openings (Figure 52, also see Section 6.7).

This feature is useful as a preview to visually verify the mapping of building walls and wall openings.

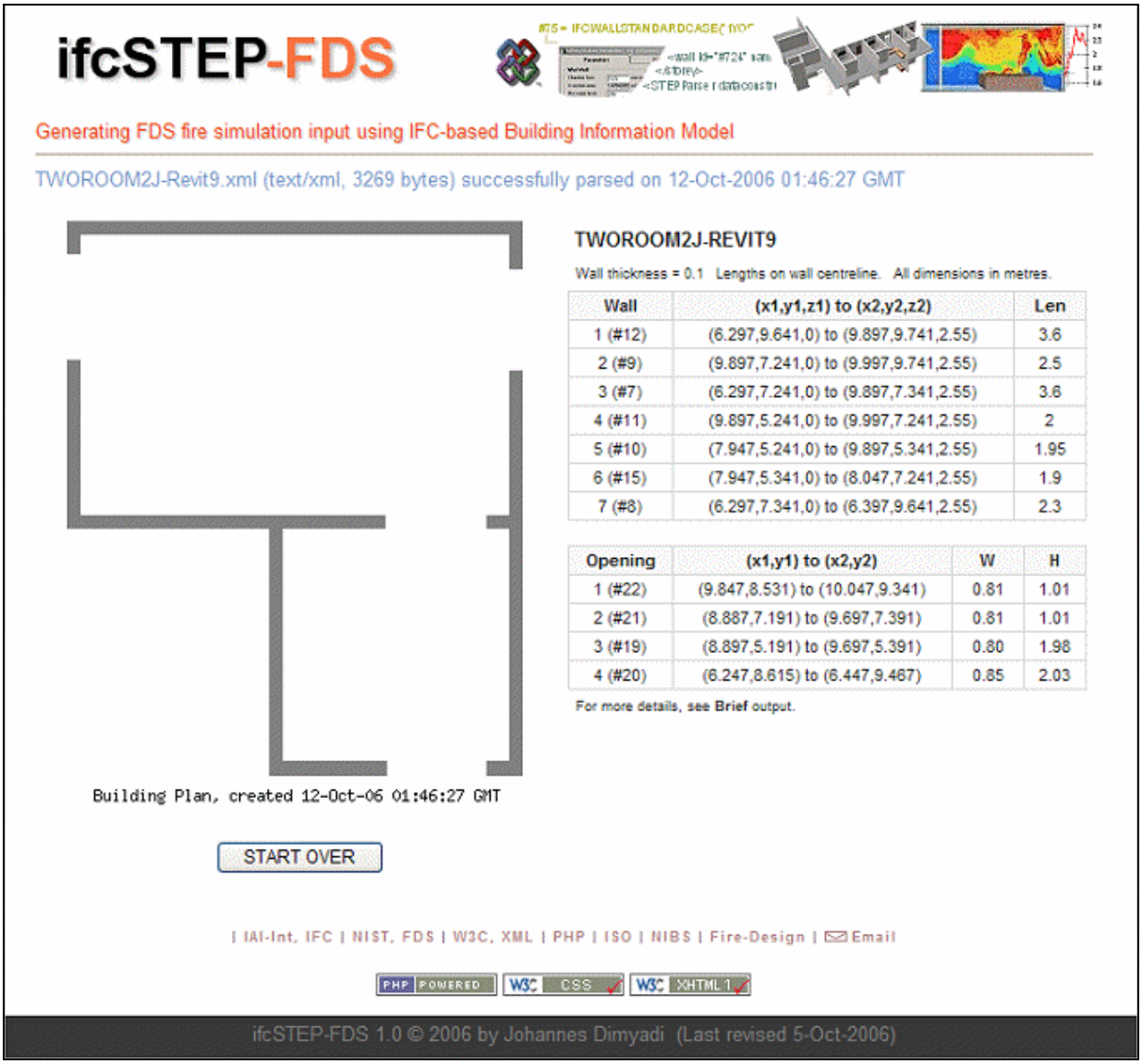


Figure 52: Screenshot of ifcSTEP-FDS Geometry Output

6.9.4 Generated FDS Input Data

This feature option parses the specified ifcSTEP Parser's compliant XML document and displays on-screen the generated FDS input data (Figure 53 & Figure 54). The displayed output text can be selected and copied to the Window's clip-board for transfer to a text file.

Figure 53: Screenshot of ifcSTEP-FDS generated FDS Input

Once transferred and saved as an external file, the generated FDS input data can immediately be fed into FDS software package. As the default duration of the simulation is set to zero in the input file, the result of the simulation would initially be the SmokeView file containing the building geometry, which can be used to preview and verify the model.

The complete output text of the generated FDS input data for a test case model is given in Figure 54.

```
&HEAD CHID='TWOROOM2J-REVIT9', TITLE='TWOROOM2J-REVIT9' /
================================================================================
FDS input data generated on 24 September 2006 05:25 by ifcSTEP-FDS © 2006 by Johannes Dimyadi.
Using TWOROOM2J-Revit9.xml document generated by ifcSTEP Parser at
14:03:38 on Friday 15 September 2006 by Michael Spearpoint.
(ifcSTEP Parser © 2005-2006 by Michael Spearpoint).
================================================================================
Change simulation duration to suit
&TIME TWFIN=0.0 /

Change the FDS database path to suit.
&MISC SURF_DEFAULT='GYPSUM BOARD', DATABASE='f:\nist\fds\database4\database4.data'
REACTION='WOOD', TMPA=20., TMPO=15. /

****** GRID DIMENSIONS (single mesh system)
Change grid dimensions to suit.
&GRID IBAR=135, JBAR=108, KBAR=24 /349920 cells total
&PDIM XBAR0=6.297, XBAR=9.997, YBAR0=5.241, YBAR=9.741, ZBAR0=0, ZBAR=2.55 /

****** DOMAIN BOUNDARIES
&VENT CB='XBAR0', SURF_ID='OPEN' /
&VENT CB='XBAR', SURF_ID='OPEN' /
&VENT CB='YBAR0', SURF_ID='OPEN' /
&VENT CB='YBAR', SURF_ID='OPEN' /
&VENT XB=6.297,9.997,5.241,9.741, 0.00, 0.00, SURF_ID='CARPET' / CARPET FLOOR
&VENT CB='ZBAR', SURF_ID='PAINTED_GIB' / PAINTED GIB CEILING

****** PRESCRIBED DESIGN FIRES
Change prescribed design fire and heat release rates to suit.
&SURF ID='FIRE_STEADY', HRRPUA=2000.0 / STEADY FIRE WITH HRR OF 2000 kW/sqm
&SURF ID='FIRE_GROWING', HRRPUA=2000.0, RAMP_Q='RAMP_FIRE' /
&RAMP ID='RAMP_FIRE', T=0.0, F=0.0 /
&RAMP ID='RAMP_FIRE', T=100.0, F=1.0 /

****** OUTPUT SPECIFICATION
Add or modify output specification as required.
ISOF QUANTITY='MIXTURE_FRACTION', VALUE(1)=0.05, VALUE(2)=0.001 / Flame and Smoke
SLCF PBY=0.0, QUANTITY='VELOCITY' /
SLCF PBX=0.0, QUANTITY='TEMPERATURE', VECTOR=.TRUE. /
BNDF QUANTITY='HEAT_FLUX' /

****** PRESCRIBED SURFACE TYPES
&SURF ID='CARPET'
 FYI='Ohlemiller cone data'
 RGB = 0.60,0.80,1.00
 BACKING = 'INSULATED'
 TMPIGN = 290.
 KS = 0.16
 C_P = 9.0
 DENSITY = 750.
 DELTA = 0.006
 FUEL_FRACTION = 0.62
 BURNING_RATE_MAX = 0.05
 HEAT_OF_COMBUSTION = 22300.
 HEAT_OF_VAPORIZATION = 2000. /

&SURF ID='PAINTED_GIB'
 FYI = 'Eurefic data BRANZ'
 RGB = 0.80,0.80,0.70
 KS = 0.17
 C_P = 0.9
 DENSITY = 731.
 DELTA = 0.010
 HRRPUA = 225.
 RAMP_Q = 'EUREFIC'
 BURN_AWAY = .TRUE.
 EMISSIVITY = 0.88
 TMPIGN = 478. /
&RAMP ID='EUREFIC',T= 0.0,F=0.0 /
&RAMP ID='EUREFIC',T= 2.0,F=0.44 /
&RAMP ID='EUREFIC',T= 4.0,F=1.0 /
&RAMP ID='EUREFIC',T= 7.0,F=1.0 /
```

```
&RAMP ID='EUREFIC',T=20.0,F=0.0 /
&RAMP ID='EUREFIC',T=30.0,F=0.0 /

&SURF ID = 'UPHOLSTERY'
 FYI = 'Fleischmann and Chen, 100% acrylic'
 C_DELTA_RHO = 1.29
 TMPIGN = 280.
 DENSITY = 40.0
 RGB = 0.53,0.38,0.35
 BURN_AWAY = .TRUE.
 BURNING_RATE_MAX = 0.03
 HEAT_OF_VAPORIZATION = 1500.
 HEAT_OF_COMBUSTION = 30000. /

****** OBSTRUCTIONS - WALLS
Wall thickness of 0.1 m has been assumed throughout
&OBST XB=6.297,9.897,9.641,9.741,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#12
&OBST XB=9.897,9.997,7.241,9.741,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#9
&OBST XB=6.297,9.897,7.241,7.341,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#7
&OBST XB=9.897,9.997,5.241,7.241,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#11
&OBST XB=7.947,9.897,5.241,5.341,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#10
&OBST XB=7.947,8.047,5.341,7.241,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#15
&OBST XB=6.297,6.397,7.341,9.641,0,2.55 SURF_ID='PAINTED_GIB' / Wall id=#8

****** WALL OPENINGS
&PERMIT_HOLE = .TRUE. /
&HOLE XB=9.847,10.047,8.531,9.341,0.9,1.91 / Opening id=#22
&HOLE XB=8.887,9.697,7.191,7.391,0.9,1.91 / Opening id=#21
&HOLE XB=8.897,9.697,5.191,5.391,0.15,2.13 / Opening id=#19
&HOLE XB=6.247,6.447,8.615,9.467,0.15,2.18 / Opening id=#20

****** PRESCRIBED BURNER OR FUEL PACKAGES
Disable the VENT line, or enable OBST line, as required.
&OBST XB=8.206,8.506,6.129,6.429,0.148,0.448 SURF_ID='UPHOLSTERY' / 0.3m x 0.3m box
&VENT XB=8.306,8.406,6.229,6.329,0.448,0.448 SURF_ID='FIRE_GROWING' / Fire on box
OBST XB=8.206,8.506,6.129,6.429,0.148,0.448 SURF_IDS='FIRE_STEADY', 'INERT', 'INERT' / A burner
```

Figure 54: Generated FDS input data

Apart from the wall obstructions and wall openings, there are also several other predefined parameters included in the generated FDS input data, as follows:

1. Prescribed Design Fires.
   There are two cases of predefined design fires included, namely Fire_Steady and Fire_Growing. In both cases, the initial Heat Release Rates per Unit Area (HRRPUA) specified is 2000 kW/m^2. Growing fire parameter is defined by a simple ramping function (Figure 54).

2. Prescribed surface material definitions.
   Three types of surface material definition and properties are included, namely Carpet, Painted_Gib, and Upholstery. The Carpet and Upholstery properties are taken from the FDS database, and the Painted_Gib is based on Eurefic cone calorimeter tests [Wade, 2003].

These predefined parameters are included by default for convenience and can be edited to suit using any text editors prior to the fire simulation.

When exploring a more conservative fire scenario, it is often desired to allow more smoke to flow into a critical area while restricting the flow into other spaces. This may be achieved by keeping certain doors and windows closed to prevent smoke from entering the space. To close a wall opening, the appropriate wall opening definition in the FDS input data can be disabled (by removing the “&” prefix) or the T_CREATE or T_REMOVE parameters can be added to the line.

7.0 Test Case Models

In order to verify the mapping implementation from Revit Buildingto ifcSTEP Parser and to ifcSTEP-FDS, a set of test case models have been created as shown in Table 7. These test case models have been constructed and used in the project to verify the exchange of wall geometry as well as opening sizes and their positions from the original model to FDS.

The test case models have all been constructed in Revit Building 9 software package with the following common parameters and constraints:

1. A topographical surface entity has been specifically created and included for exchange as *IfcSite* entity as required by the ifcSTEP Parser.

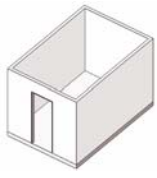
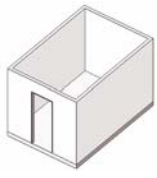
2. Due to the current limitations of ifcSTEP Parser, all walls have been constructed upright, equal height and orthogonal with respect to each other and have been given a constant thickness throughout.

3. As *ifcSpace* entiry is not required for the generation of objects in FDS, no space/room objects have been specifically created or included in the models.

4. Door and window components included in Revit Building software package have been used. Wall openings have been created using the wall cut-out modelling tool.

5. All models consist of only one storey.

The IFC2x2 compliant STEP files for each of the test case models were obtained by exporting directly from the Revit Building 9 software package. These files were then parsed by ifcSTEP Parser software application to create the intermediate XML documents for use by ifcSTEP-FDS.

#	Model Name	Model Description	Model 3D View
1	**Single Room ISO9705 A**	Single room 3.6m x 2.4m and 2.4m high used in ISO 9705 fire test with a single 800x2000 door. Walls are all attached to the floor slab.	
2	**Single Room ISO9705 B**	Same as ISO9705 A, except that the door has been replaced with a wall opening of the same size. Walls are all attached to the floor slab.	

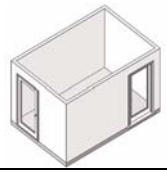
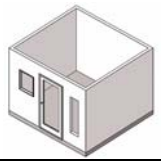
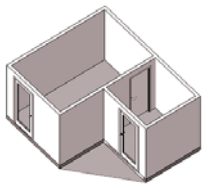
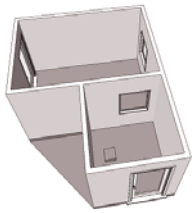
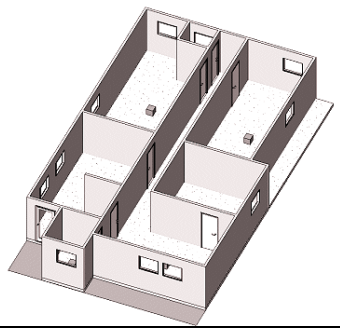
3	**Single Room 2**	Single room 3.5m x 2.3m and 2.4m high. Two doors with different swing directions on the external walls	
4	**Single Room 3**	Single room building with different types of opening, i.e. a window, a door and a plain opening, on an external wall.	
5	**Single Room 5A**	Single room building with a window and a plain opening on one wall, A door and a window on another wall. Walls have no bottom constraints.	
6	**Single Room 5B**	Same as Single Room 5A, except that the walls are attached to the floor. A box is also included in the middle of the room in this model.	
7	**Two-Room D**	Two-Room building with two external doors and one internal door, different swing directions.	
8	**Two-Room E**	Two-Room with a plain opening on an external wall, a door on an external wall, a window on an external wall, and a window on an internal wall.	
9	**Two-Room J**	Two-Room with two external doors, one external window and one internal window.	
10	**Multi-Room 2**	Multi-room with a long corridor. A door and a series of windows on external walls. An internal door to each room. A furniture item in the centre of each room. Walls have no top or bottom constraints.	

Table 7: Test Case Models

The geometry for each of the test case models together with the 3D view of the model in Revit Building as well as the resulting view in Smokeview are given in the Appendix 2.

8.0 Scope and Limitations

8.1 Scope of Model Geometry

The scope of the model geometry that can be mapped to FDS as the result of this work is summarised below:

1. Single storey building.
2. Source model must contain *IfcProject, IfcSite*, and *ifcBuildingStorey.*
3. Walls are assumed to have a constant height.
4. Walls have uniform cross sections as well as longitudinal sections.
5. Wall paths must be straight lines and parallel to X and Y axes, hence orthogonal with respect to each other
6. Wall vertical axis must be parallel to the Z axis, i.e. upright and orthogonal with respect to the floor and ceiling planes.
7. Walls must have a constant thickness.
8. Walls cut-outs or openings without the associated doors or windows must be detached from the floor to ensure that it is exchanged as *IfcOpeningElement.*
9. At least one wall in the building must be exchanged as *IfcWallStandardCase* entity.
10. Placement origin of walls must be on the centreline of the wall.
11. Placement origin of door, window and wall openings must also be on the centreline of the wall.
12. Types of wall body representation supported are limited to Swept Solid and Clipping.
13. BREP geometry is not supported.
14. Only the placement origin of furniture items is exchanged. This is assumed to be at the centre of the base of the furniture item.
15. Wall material type is currently ignored.
16. Floor slab and ceiling entities are not mapped. However, the finished floor plane is taken to be at the same level as the base of the wall and the ceiling plane is at the same level as the top of the wall.
17. Door, window and wall openings are all exchanged as full openings through the walls. Partial opening is not supported.

8.2 Implementation Related Issues

Some of the implementation related issues observed during the project can be summarised as follows:

1. The test case models were initially constructed in Revit Building version 8.1, which was only partially certified for IFC (stage 1 certification). There were some inconsistencies observed in the implementation of the local placement coordinates for the door and window openings. This has subsequently been rectified in the next version of Revit Building, which was used primarily for the remainder of the project.

2. Models created in Revit Building older than version 9 may not translate as expected when the application is upgraded to Version 9. It appears that certain door objects created in the old model would not inherit the new set of properties available in the new version of the application. For correct mapping, door objects in the old model must be removed and replaced with the new objects that are supplied or constructed in Revit Building 9.

3. By default, Revit Building 9 places wall objects on the centreline. This may not be the case for other BIM authoring software packages such as ArchiCAD.

4. A topographical element must be specifically created in Revit Building in order for the required *IfcSite* entity to be created as part of the STEP export.

8.3 ifcSTEP Parser Limitations

ifcSTEP Parser has been revised to incorporate the new set of information required for mapping standard walls and wall openings to FDS. However, the current scope of its IFC data mapping implementation is still somewhat limited. The main limitations can be summarised as follows:

- No support for BREP geometry.

- Only straight, orthogonal and equal height walls can be mapped. Cannot parse diagonal, inclined and curved surfaces.

- Mapping of building elements are limited to standard walls, wall openings, doors, windows, roof and floor slabs. Horizontal openings in floor & roof and stair elements are not supported.

- IFC property sets are not supported, which limits the mapping of properties to only those already hard-coded as attributes in the IFC model. The support for property sets may allow other more specific properties such as the heat release rates of fire, species yields, heat of combustion, etc. to be attached to the model and exchanged like any other attributes.

- Only limited attributes of floor and roof slabs are mapped. Position coordinates are not provided.

- Mapping of a furniture item is limited only to its placement origin.

- IFC support is limited to version 2x2.

8.4 ifcSTEP-FDS Limitations

Apart from the limitations inherited from ifcSTEP Parser, there are other limitations specific to ifcSTEP-FDS, which can be summarised as follows:

- The current implementation only supports single storey building.
- Only 2D building plan geometry can be generated for display.
- The FDS computational grid resolution is given as a default value without optimisation. User must amend the resolution to suit each simulation as desired.
- No algorithms for stair-stepping of diagonal, inclined or curved surfaces have been implemented.
- The generated FDS input data must first be copied and saved to an external file before it can be edited using text editors.
- Design fires and other simulation parameters are provided by means of prescribed data.
- One limitation for being a web application is that FDS and Smokeview cannot be directly executed from ifcSTEP-FDS.
- No other import facilities have been implemented apart from the initial XML input document.

9.0 Future Work

9.1 IFC-STEP Parser

A list of potential areas that could further be incorporated into the scope of the data mapping implementation in ifcSTEP Parser would include the following:

- BREP geometry.
- Mapping of diagonal and curved walls.
- Mapping of horizontal openings, stairs and other building elements.
- Mapping of fire protection systems, e.g. location of sprinkler heads and detectors.
- Incorporate a set of specific fire engineering property that can be incorporated into BIM models.
- Mapping the centroid or bounding box of furniture items based on the placement origin and the overall dimension.
- Wall fire rating properties.
- Explore the differences in IFC export implementation of different BIM authoring tools.

There are also a host of new entities that may be explored and incorporated based on the latest version of IFC 2x3.

9.2 IfcSTEP-FDS

As ifcSTEP-FDS is an interface application to ifcSTEP Parser, further development work in ifcSTEP-FDS would depend on the scope of information provided in the XML output document, which may include:

- Incorporation of an algorithm to create stair-stepping of diagonal, inclined and curved surfaces.
- Ability to view dynamic 3D geometry of the building model in VRML [ISO/IEC 14772, 1997] or X3D [ISO/IEC 19775, 2004] formats, instead of just the static 2D building plan.
- Mapping of floor slabs, roof elements, and stairs for multi-storey buildings.
- Incorporation of an algorithm to optimise the size of FDS computational domain based on the building geometry.
- Mapping of fire engineering specific property sets.
- Development of a new visual model editor feature as part of the application, which would import FDS input data for editing.
- Mapping of fire protection systems, e.g. location of sprinkler heads and detectors, etc.

10.0 Conclusion

The project has reviewed the way in which CAD has been used in the design of building and to what extent the information can be shared by the project stakeholders. It also looked at the advantages of BIM in comparison with the conventional CAD and the way in which the information can be exchanged using the IFC data model.

The project has also reviewed a software utility, ifcSTEP Parser, which has been developed specifically to exchange IFC data with fire simulation applications. Some changes and extension have been proposed in this project to widen the scope of its IFC data mapping implementation.

A web application, ifcSTEP-FDS, has been developed in this project as an interface to ifcSTEP Parser. The purpose of the application is to take the XML output from the ifcSTEP Parser and generate the input data for FDS.

IfcSTEP-FDS has successfully been tested in conjunction with the revised version of ifcSTEP Parser to convert a number of test case BIM models and to generate the corresponding FDS input data. Given the complexity of the IFC data model and the variants in the BIM modelling methods to be explored within the project time frame, the data mapping implementation of the applications is currently limited to the basic building geometry. However, there is a potential scope for further work, which has been highlighted in the report.

The AEC/FM industry has recognised the benefits that can be gained from sharing common digital building information from the design phase through the facilities management and to the end of the building life-cycle. The industry standard model for interoperability has now been made available through the IFC technology. There are certainly incentives in the industry to move towards IFC interoperability, so it is now up to the software vendors and the building designers/managers to make use of the technology to achieve the common goal.

While there has been a growing interest in the other domains in the AEC/FM industry to use the IFC data model, particularly for downstream applications, there is certainly a scope in the fire engineering domain that could benefit from this. The initiatives taken by ifcSTEP Parser and subsequent interface applications would potentially provide a considerable cost saving and improvement on the quality of data exchanged for fire simulations and fire safety design in general.

11.0 References

ActiveFacility (2005), http://www.activefacility.com, (accessed 27 June 2006).

Adachi, Y (2001), "IFC 2x Property Set Development Guide" (Draft 2). Model Support Group. International Alliance for Interoperability. June 2001.

AIA Building Connections, "National Institute of Standards and Technology – Data Exchange Standards Activity", http://www.building-connections.info/organizations/nist.html (accessed 19 Dec 2006).

Amor, Robert (1997), "Product Models in Design and Engineering". Building Research Establishment Limited, Garston, Watford WD2 7JR, UK, 1997.

Autodesk (2005), "Building Information Modelling: A Key to Performance-Based Design" (White Paper), (accessed 17 July 2006).

Brown, Bob (2002), "A Gentle Introduction to PHP". School of Computing and Software Engineering, Southern Polytechnic State University, USA, http://www.spsu.edu/cs/faculty/bbrown/papers/php1.html, (accessed 20 May 2006).

BCA (2005), Building and Construction Authority (BCA) of the Republic of Singapore, http://www.bca.gov.sg/e-services, (accessed 27 June 2006).

Chiam, B H (2005), "Numerical Simulation of a Metro Train Fire". Master of Engineering in Fire Engineering Dissertation. University of Canterbury, New Zealand. June 2005.

Clement, J M (2000), "Experimental Verification of the Fire Dynamics Simulator (FDS) Hydrodynamic Model". Doctor of Philosophy in Fire Engineering Dissertation. University of Canterbury, New Zealand. October 2000.

CORENET (2005), Construction and Real Estate Network (CORENET), http://www.corenet.gov.sg, (accessed 8 August 2006).

Davis, Z (2002), "Improving Teamwork and Collaboration in Design Environments" (white paper). Ziff Davis Media.

Dimyadi, J A W (2004). "Guidelines for extracting 3DSOLID geometry in AutoCAD for the DXF to FDS Conversion Utility" (unpublished).

FIC (2006), "Developing the National Building Information Model Standard". Facilities Information Council (FIC) committees, BIM Committee, National Institute of Building Sciences (NIBS), USA, http://www.nibs.org/BIMcommittee.html, (accessed 7 July 2006)

Forney G P, McGrattan, K B (2006), “User Guide for SmokeView Version 4 – A Tool for Visualizing Fire Dynamics Simulation Data”. NIST Special Publication 1017. National Institute of Standards and Technology, USA.

Frost, I et al (2001), “A semi-automated approach to CAD input into Field Based Fire Modelling Tools". Proceedings of the 9th International Fire Science and Engineering Conference: Interflam 2001 vol. 2, pp 1421-1426, Edinburgh, Scotland.

Gallaher, M et al. (2004). “Cost Analysis of Inadequate Interoperability in the US Capital Facilities Industry” (NIST CGR 04-867). National Institute of Standards and Technology, US Department of Commerce, USA.

Graphisoft (2003), "Graphisoft strengthens commitment to Interoperability & International IFC Industry Initiative", White Paper, March 2003.

IAI (2006a), International Alliance for Interoperability, http://www.iai-international.org/About/History.html, (accessed 27 June 2006).

IAI (2006b), International Alliance for Interoperability, International Overview of IFC-Implementation Activities. http://www.bauwesen.fh-muenchen.de/iai/ImplementationOverview.htm (accessed 17 Sep 2006).

IAI-NA (2006), “Mission Statement”, International Alliance for Interoperability - North America Chapter, http://www.iai-na.org/about/

Interop AEC+fm (2001), “Industrial Use of the Building Modelling Approach”., Sydney, Australia 2001.

ISO 9705-33 (1993), “Fire tests on building materials and structures - Part 33. Full-scale Room Test for Surface Products”. International Organization for Standardization.

ISO 10303-11 (2004), “Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 11: Description Methods: The EXPRESS Language Reference Manual”. International Organization for Standardization.

ISO 10303-21 (2002), “Industrial Automation Systems - Exchange of Product Model Data - Part 21: Implementation Methods; Clear Text Encoding of the Exchange Structure”. International Organization for Standardization.

ISO 10303-22 (1998), “Industrial Automation Systems - Exchange of Product Model Data - Part 22: Implementation Methods; Standard Data Access Interface”. International Organization for Standardization.

ISO 10303-28 (2003), “Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 28: Implementation Methods: XML Representations of EXPRESS Schemas and Data”. International Organization for Standardization.

ISO 10303-42 (2003), “Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 42: Integrated Generic Resource: Geometric and Topological Representation”. International Organization for Standardization.

ISO 10303-43 (2000), “Industrial automation systems and integration - Product Data Representation and Exchange - Part 43: Integrated Generic Resource: Representation Structures”. International Organization for Standardization.

ISO/IEC 14772-1 (1997) and ISO/IEC 14772-2 (2004), “Virtual Reality Modelling Language (VRML)”. International Organization for Standardization.

ISO/IEC 19775 (2004), “Extensible 3D (X3D)”. International Organization for Standardization.

Jensen, Ken (2003), “TheLaiserinLetterLetters”. The Laiserin Letter, Issue 18, 13th January 2003, http://www.laiserin.com/features/issue18/feature04.php

Kay, Rana (2006), “Building Information Modelling adds new dimension to architectural design”. San Diego Daily Transcript. Architecture Week. San Diego, USA.

Khemlani, Lachmi (2004a). "What's in a Name? The BIM Acronym". AECbytes Newsletter #5 (5th February 2004). (last accessed 1 July 2006).

Khemlani, Lachmi (2004b). "The IFC Building Model: A look under the hood" AECbytes Feature (30th March 2004). http://www.aecbytes.com/review/2006/ArchiCAD10.html (last accessed 1 July 2006).

Khemlani, Lachmi (2005a). “CORENET e-PlanCheck: Singapore’s Automated Code Checking System”. AECbytes "Building the Future" Article (26th October 2005). http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html (last accessed 27 July 2006).

Khemlani, Lachmi (2005b). “Multi-Disciplinary BIM at Work at GHAFARI Associates”. AECbytes Features Article (21st November 2005). http://www.aecbytes.com/feature/2005/Ghafari_study.html (last accessed 24 October 2006).

Khemlani, Lachmi (2006). AECbytes "ArchiCAD 10" AECbytes Product Review (31st July 2006). http://www.aecbytes.com/review/2006/ArchiCAD10.html (last accessed 3 September 2006).

Liebich, T., and Wix J., editors. (2000), IFC technical guide, Industry Foundation Classes – Release 2x, International Alliance for Interoperability.

Liebich, Thomas (2001). “ifcXML language binding of EXPRESS”. Version 1.02, 11 December 2001. International Alliance for Interoperability.

Liebich, Thomas (2002). “Standard Analysis – Current AEC Situation – Building Models”. Final Report. ProdAEC project IST-2001-32035. 20 August 2002.

Liebich, Thomas (2004). “IFC 2x Edition 2 Model Implementation Guide”. Version 1.7, 18 March 2004. International Alliance for Interoperability.

Liebich, T et al (2006), “IFC 2x Edition 3”. International Alliance for Interoperability.

McGrattan, K B, editor (2005a), “Fire Dynamics Simulator (Version 4) Technical Reference Guide”. NIST Special Publication 1018. National Institute of Standards and Technology, USA.

McGrattan, K B, Forney G P (2005b), “Fire Dynamics Simulator (Version 4) User’s Guide”. NIST Special Publication 1019. National Institute of Standards and Technology, USA.

Moghaddam, A Z et al (2004), “Fire Behaviour Studies of Combustible Wall Linings Applying Fire Dynamics Simulator”. Centre for the Environmental Safety and Risk Engineering. Victoria University of Technology. Paper presented at 15th Australasian Fluid Mechanics Conference at the University of Sydney, Australia.

NIST (2006), “DXF2FDS Documentation”, NIST Fire Dynamics Simulator (FDS) and Smokeview. http://fire.nist.gov/fds/

Peacock, R et al (2005), “CFAST – Consolidated Model of Fire Growth and Smoke Transport (Version 6) – User’s Guide”. NIST Special Publication 1041. National Institute of Standards and Technology, USA

Spearpoint, M J (2003), “Integrating the IFC building product model with zone fire simulation software”. Proceeding International Conference on Building Fire Safety, QUT, Gardens Point Campus, Brisbane, Australia, 20-21 November 2003, pp. 56-66

Spearpoint, M J (2005), “Transfer of architectural data from the IFC model to a fire simulation software tool”. Accepted for publication by the Journal of Fire Protection Engineering.

Stangeland, B (2005). “Efficient flow of Information in the building process using IFC”. Paper presented at the IFC Conference in Oslo, Norway, 1st June 2005.

Thompson, P A (1995). “A computer model for the evacuation of large building populations”. Fire Safety Journal vol.24 pp 131-148.

Thunderhead Engineering Consultants (2006), “PyroSim User Manual” (2006.2). Thunderhead Engineering Consultants in collaboration with The RJA Group Incorporated.

W3C (2006), “HyperText Markup Language (HTML)”, http://www.w3.org/markup/, (accessed 25 June 2006).

Wade C A (2003), “A user’s guide to BRANZFIRE 2003”. Building Research Association of New Zealand. Judgeford, Porirua City, New Zealand.

Walker, J (1994), “The Autodesk File. Bits of history, words of experience”. Fourth edition. http://www.fourmilab.ch/autofile/www/autoframe.html (accessed 14 October 2006).

Wikipedia contributors (2006), "Aperture (CAD software)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Aperture_%28CAD_software%29&oldid=80189849 (accessed October 27, 2006)

Wix, J (1998), “IFC Specifications Development Guide” (Revision Beta.d2 Jan 1999). Specification Task Force. International Alliance for Interoperability.

Appendix 1: Keywords, Abbreviations and Definitions

ADO.NET	ADO is an acronym for ActiveX Data Objects, which is a new generation of OLE (Object Linking and Embedding) developed by Microsoft Corporations. It is a set of computer software components that can be used by programmers to access data and data services. ADO.NET is a heavily revised ADO technology. More information can be obtained from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/adonetanchor.asp
AEC/FM	An acronym for Architectural, Engineering, Construction and Facilities Management.
Apache	An open source HTTP server (see Web-Server) developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Particularly popular among hosting companies supporting PHP and MySQL. More information: http://www.apache.org/
BIM	An acronym for Building Information Modelling. An object-oriented computer aided design concept in a 3-dimensional integrated environment that can be used to assemble various building components and maintain a virtual model of the building. More information can be obtained from http://www.nibs.org/BIMcommittee.html
BRANZFIRE	The name of a zone-model fire simulation software application developed by BRANZ (Building Research Associations of New Zealand) [Wade, 2003]
BuildingSMART	A concept initiated by the North American Chapter of the IAI and developed by the US National Institute of Building Sciences (NIBS). It is a standard universal framework to enable and encourage information sharing and interoperability throughout all phases of the whole building life cycle. More information can be obtained from http://www.buildingsmart.org.au/
C, C++, C#	A general purpose programming language developed in the early 1970s by Dennis Ritchie at AT&T Bell Lab for use on the Unix operating system. It is currently one of the most widely used programming languages.
CAD	An acronym for Computer-Aided Design. Historically, the Euclidean geometry invented by the mathematician Euclid of Alexandria in 350 BC has been the foundation upon which modern CAD systems have been developed. In the context of AEC/FM industry, it refers to a genre of computer application for the design and documentation process.
CFD	An abbreviation for Computational Fluid Dynamics. It refers to the use of computers to analyse problems in fluid dynamics.

Client-side	Operations performed by the client in a client-server network such as the internet. Typically, a client is a computer application such as a web browser that runs on the user's local computer. See Server-side.
Cross-platform	Another term for platform-independent. It refers to software applications that do not use any features specific to a particular computer system, operating system or programming language.
CSS	An abbreviation for Cascading Style Sheet. It is a style-sheet language used to describe the presentation of a document written in a mark-up language such as HTML. More information can be obtained from http://www.w3.org/Style/CSS/
Cuboid	A geometrical object in the shape of a rectangular box bounded by six rectangular faces perpendicular to each other. It is equivalent to a right parallelepiped.
DataSet	In the context of ADO.NET, it is a temporary object that represents the whole database in the computer memory. It can contain tables as well as relationships between the tables of the database. More information can be obtained from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbcondatasets.asp
DXF	An abbreviation for Drawing Exchange Format. A data format in either ASCII or binary formats originally developed by Autodesk in 1982 as a solution for data interoperability between AutoCAD and other software applications. It was intended to provide an exact representation of the data in the AutoCAD proprietary DWG format. More information can be obtained from http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=5129239
Expat Library	Expat is an open-source XML parser toolkit and code library written in C in 1998 by James Clark, the XML working group technical leader at the World Wide Web consortium (W3C). More information can be obtained from http://expat.sourceforge.net/
FDS	An abbreviation for Fire Dynamics Simulator. A fire simulation software application based on a field or CFD model developed by NIST (US National Institute of Standards and Technology) [McGrattan, 2005b].
Fortran 90	Fortran is an acronym derived from its original name Formula Translating System. It is a general purpose programming language developed by IBM in the 1950s and has been used extensively in scientific and engineering applications for decades. Fortran 90 is one version of the language. The current version is Fortran 2003. More information can be obtained from http://www.nag.co.uk/sc22wg5/

GD Graphics Library	Originally, an abbreviation for GIF Draw. It is an open source code library written in C by Thomas Boutell in 1993 for the dynamic creation of images by programming. More information can be obtained from http://www.boutell.com/gd/
HTML	An abbreviation for HyperText Mark-up Language. A subset of Standard Generalised Mark-Up Language (SGML) developed by W3C Consortium for electronic publishing, specifically for creating hypertext documents for the World Wide Web.
HTTP	An abbreviation for HyperText Transfer Protocol. A method used to transfer or convey information on the World Wide Web. It is a patented open internet protocol coordinated by the W3C Consortium originally intended to provide a way to publish and receive HTML pages.
IFC	An abbreviation for Industry Foundation Classes. An ISO endorsed open standard object-oriented model developed by International Alliance for Interoperability (IAI) for the AEC/FM industry to share common building information.
Internet	A worldwide, publicly accessible network of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol (IP).
ISO	An acronym for International Organisation for Standardisation.
JavaScript	A prototype-based client-side scripting programming language loosely based on C. It is a Netscape Communications Corporation's implementation of ECMAScript, which was standardised by the European Computer Manufacturers Association (ECMA) in 1997. Javascript is now a registered trademark of Sun Microsystems Inc. More information can be obtained at http://www.mozilla.org/js/
JScript	Microsoft's implementation of the ECMAScript. It is a component based scripting language based on Active-X, which is the new generation of Object Linking and Embedding (OLE) technology developed by Microsoft in 1996.
NASA	An acronym for the US National Aeronautics and Space Administration.
NIST	An acronym for the US National Institute of Standards and Technology.
NIBS	An acronym for the US National Institute of Building Sciences.

OpenGL	An acronym for Open Graphics Library. It is a standard specification defining a cross-platform and cross-language Application Programming Interface (API) developed by Silicon Graphics Inc. in the early 1990s. It is intended as a rendering engine for used in applications that produce interactive three-dimensional computer graphics. More information can be obtained from http://www.khronos.org/opengles/
Open-source	In the context of software applications, it is a practice of making available the source code of the application under a copyright licence that permits users to study, change and improve the software, and to redistribute it in modified or non-modified form. More information can be obtained from http://www.opensource.org/
Parse	In the context of computer science, this means reading and interpreting an electronic document or an input sequence and transforming the input text into a data structure.
PHP	A recursive abbreviation for PHP Hypertext Processor. It was originally called Personal Home Page, a relatively simple server-side scripting language developed by Rasmus Lerdof in 1994. It is now an interpretive, HTML embedded, server-side scripting language very popular among web applications developers and well-supported by open-source developers world-wide. More information can be obtained from http://www.php.net
Plot3D	A common data format used by various software applications, particularly popular among applications developed and released by NASA. It usually contains CFD structured meshes and coordinates. Information about Plot3D can be found at http://www.grc.nasa.gov/www/wind/valid/plot3d.html
Server-side	Operations performed by the server in a client-server network such as the internet. Typically, the server is a web server that runs on a remote computer system. See Client-side.
SmokeView	A software application developed by NIST for visualising FDS simulations results in 3-dimensional graphics using openGL. More information can be obtained from http://fire.nist.gov/fds/
SPF	An abbreviation for STEP Physical File (ISO 10303-21). The file often has the extensions of .stp or .step. It has been given the extension .ifc by the export implementation of BIM authoring software applications.
STEP	An acronym for Standard for the Exchange of Product Model Data. It is an alternative term for ISO 10303 (Industrial automation systems and integration – Product data representation and exchange).

W3C	An abbreviation for the World Wide Web Consortium. An international consortium of member organisations, a full-time staff and the public collaborating to develop standards for the World Wide Web. More information can be found at http://www.w3.org
Web application	A software application that can be accessed with a web browser over a network such as the internet or an intranet.
Web Browser	A software application that enables a user to display and interact with the text, images and other information typically located on a web page at a website on the World Wide Web or a local area network.
Web Page	A resource of information (often in HTML and other linked objects) that is suitable for the World Wide Web and can be accessed through a web browser.
Web-Server	A computer system or a computer program that is responsible for accepting HTTP requests from and delivering web pages to web browsers.
World Wide Web	Also known as “www” or simply the “web”. It is a collection of interconnected resources or documents accessible on the internet, each identified by a short and unique global identifier called Uniform Resource Identifiers (URI). These resources are linked by hyperlinks and Uniform Resource Locators (URLs). More information can be obtained from http://www.w3.org/www/
XML	An abbreviation for eXtensible Markup Language. A standard defined by the World Wide Web consortium (W3C) to mark up documents containing structured information in a simple text format for data interchange over the internet. More information can be found at http://www.w3.org/xml

Appendix 2: Test Case Models

A2.1 Single Room Model ISO 9705

This is a basic single room test model, which is the representation of the ISO 9705 [ISO, 1993] room (Figure 55 & Figure 56) with the dimensions of 2.4m x 3.6m x 2.4m high. This is used to verify the exchange of basic wall geometry.

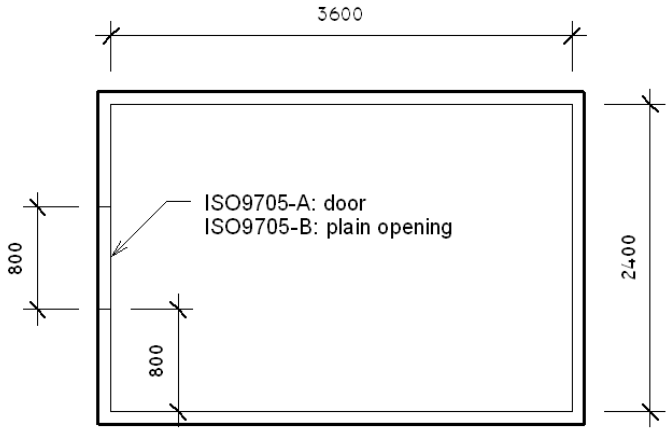
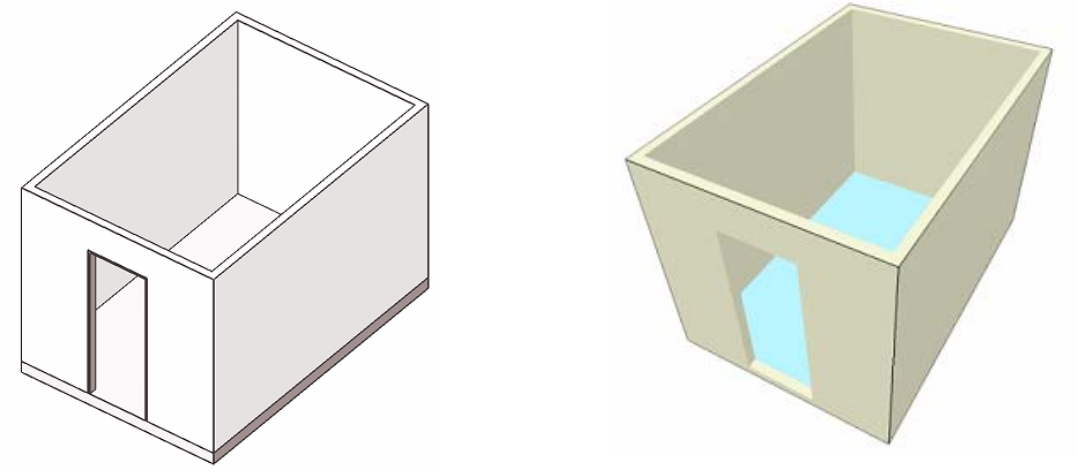


Figure 55: Floor plan of the Single Room Model ISO 9705



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 56: 3D view of the Single Room Model ISO 9705

Two versions of this model has been created, i.e. ISO9705-A with a 800x2000 doorway represented by a door object in Revit Building, and ISO9705-B has a 800x2000 wall opening created by the wall cut-out modelling tool in Revit Building.

As the walls have bottom constraints, the wall containing the opening in ISO9505-B model would be exchanged as *IfcWall* and the opening would be treated as part of the BREP profile of the wall instead of being exchanged as *IfcOpeningElement*.

By removing the bottom constraints from the wall containing the opening in the ISO9505-B model, both models have been parsed correctly and converted to FDS input accurately.

A2.2 Single Room Model 2

This is a basic single room model similar to the ISO 9705 room but with two external doors (Figure 57 & Figure 58). This model has been used to verify the exchange of doors with different "placement directions". It has been established (refer Section 5.3) that there are two scenarios for the door placement relative to the parent wall in the BIM authoring environment, i.e. +ve direction where the door is placed in the same direction as the parent wall, and –ve direction where the door is placed in the opposite direction to the parent wall.

Generally, FDS is more interested in the wall opening rather than the actual door, so the door swing direction is not of particular relevance.

This model has been correctly parsed and mapped to FDS.

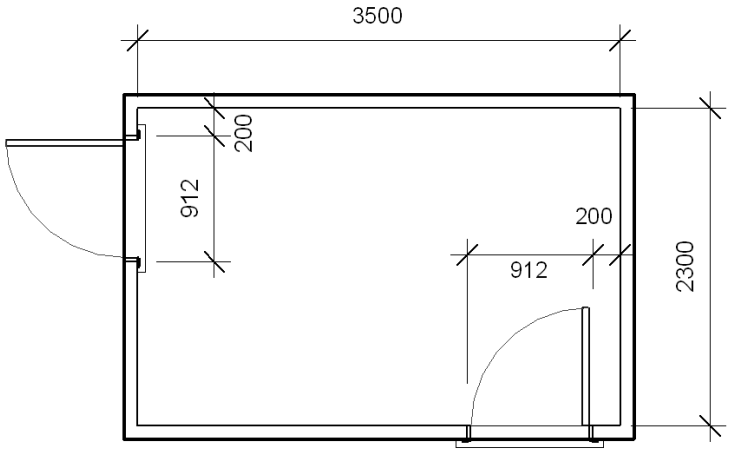
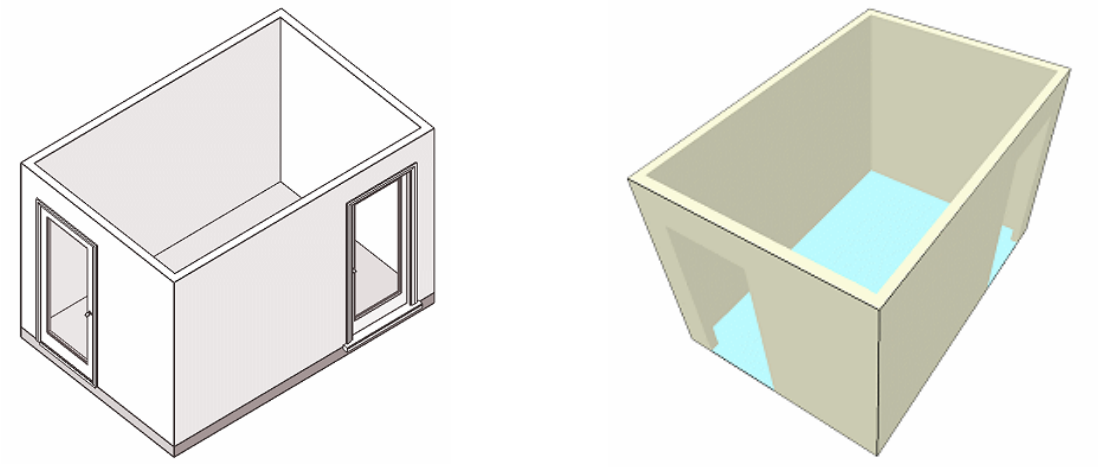


Figure 57: Floor plan of the Single Room Model 2



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 58: 3D view of the Single Room Model 2

A2.3 Single Room Model 3

This is another single room model with three different types of opening on one wall, i.e. a window, a door and a plain opening (Figure 59 & Figure 60). The model is used to observe how different types of opening are exchanged.

This model has been successfully parsed and mapped to FDS.

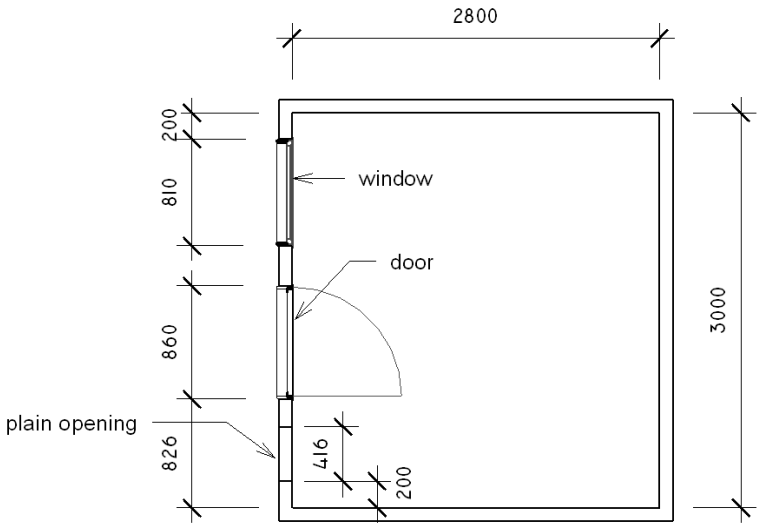
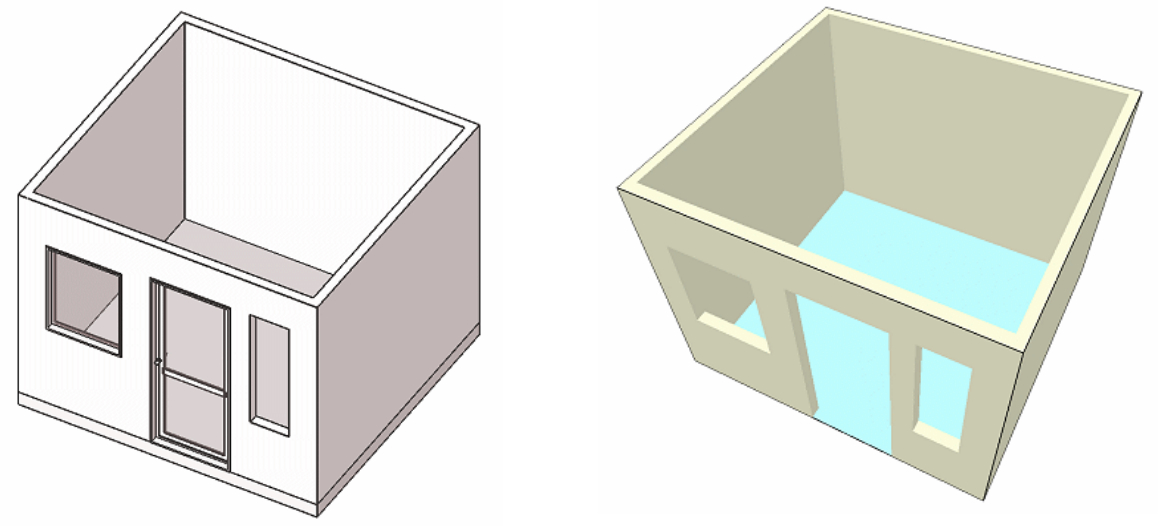


Figure 59: Floor plan of the Single Room Model 3



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 60: 3D view of the Single Room Model 3

A2.4 Single Room Model 5

This is a single room model with a window and a door on one wall as well as a window and a plain wall opening on another wall (Figure 61 and Figure 62).

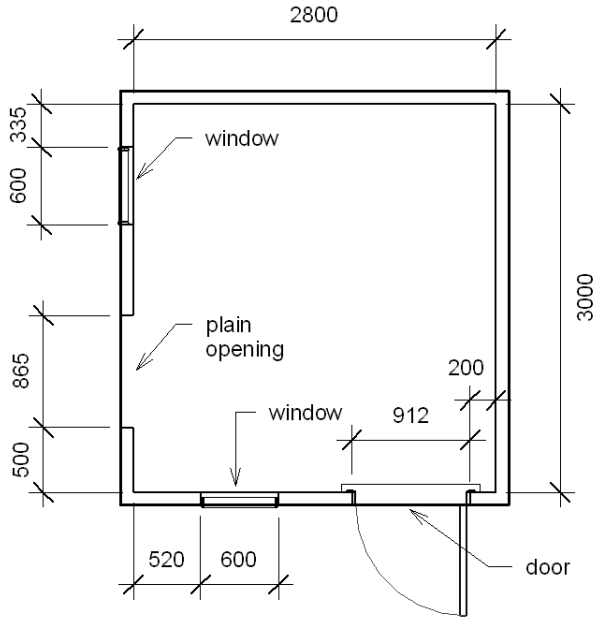
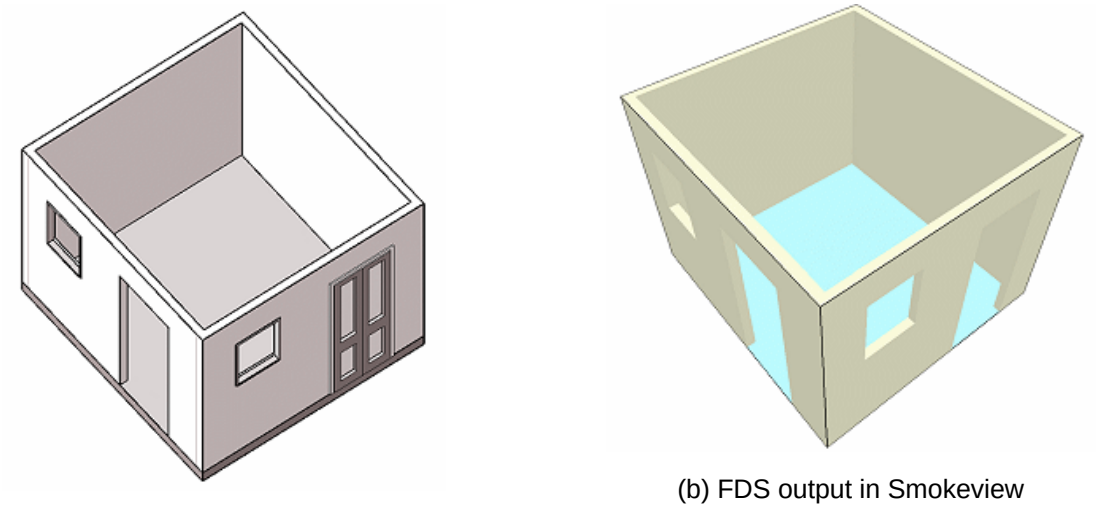


Figure 61: Floor plan of the Single Room Model 5



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 62: 3D view of the Single Room Model 5

Two versions of this model have been created, i.e. Model 5A and 5B. The only difference between them is that the wall containing the plain opening has bottom constraints in Model 5A and the constraints removed in Model 5B.

Similar to the ISO9505-B case, it has been observed that the opening in Model 5A was not exchanged as *IfcOpeningElement*, but instead it was treated as part of the BREP profile of the wall.

A2.5 Two-Room Model D

This is a two-room model with two external doors and one internal door, all having different swing directions (Figure 63 and Figure 64).

This model has been successfully parsed and correctly mapped to FDS.

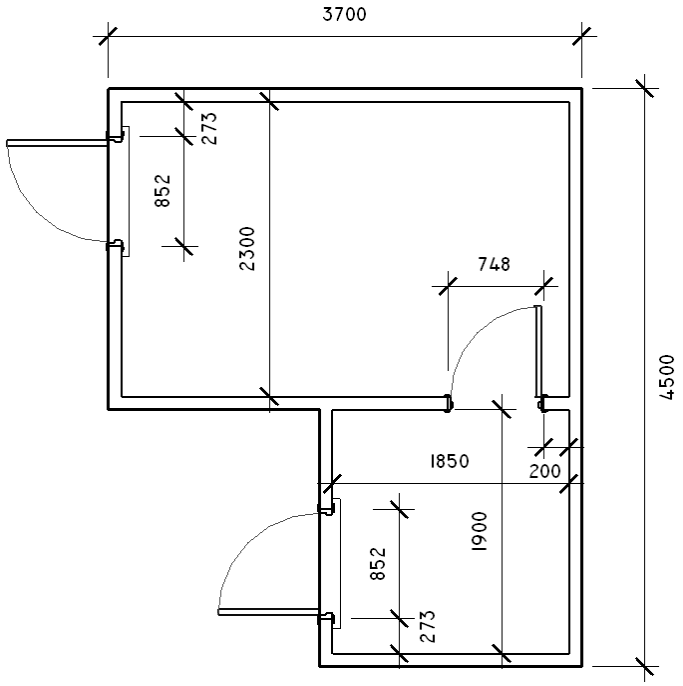
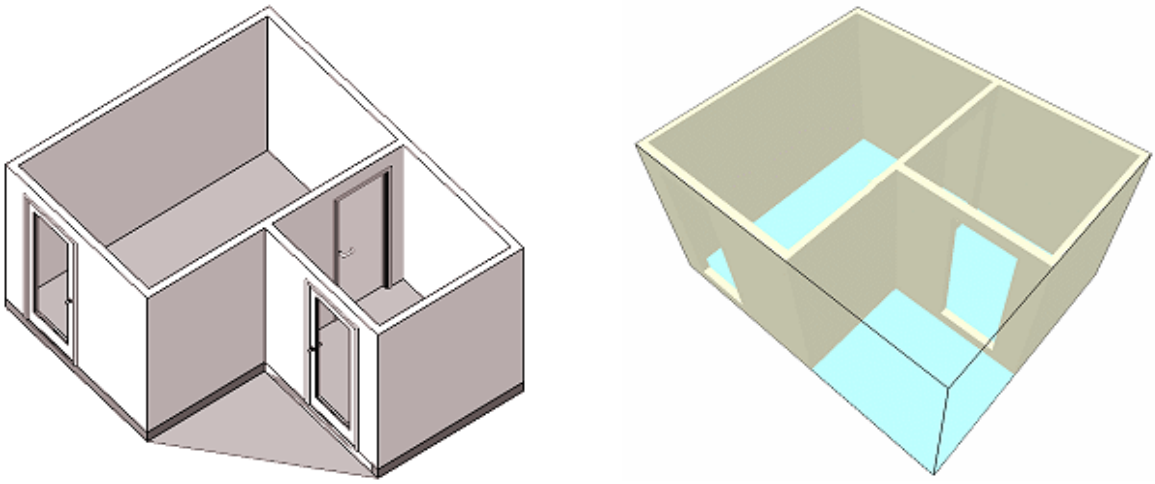


Figure 63: Floor plan of the Two-Room Model D



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 64: 3D view of the Two-Room Model D

A2.6 Two-Room Model E

This is a two room model with an external door, an external window, an external plain opening, and an internal window (Figure 65 and Figure 66).

This model has been successfully parsed and correctly mapped to FDS.

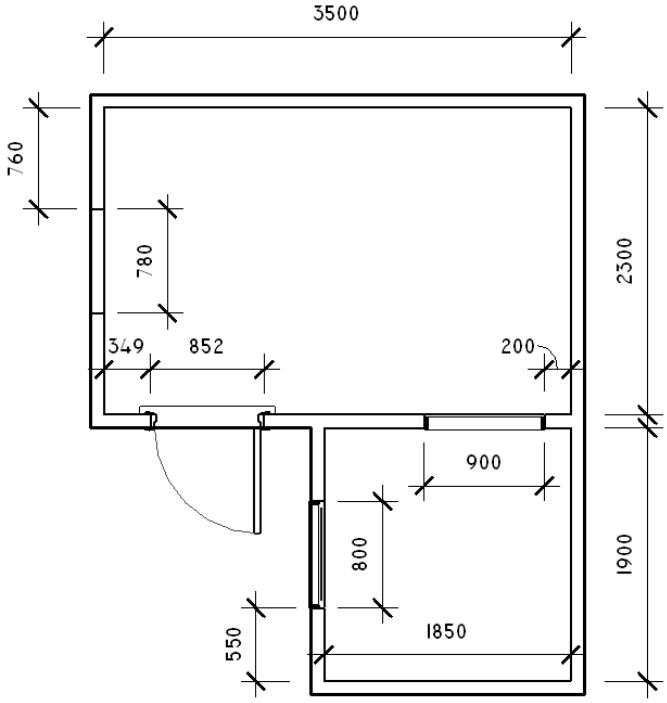
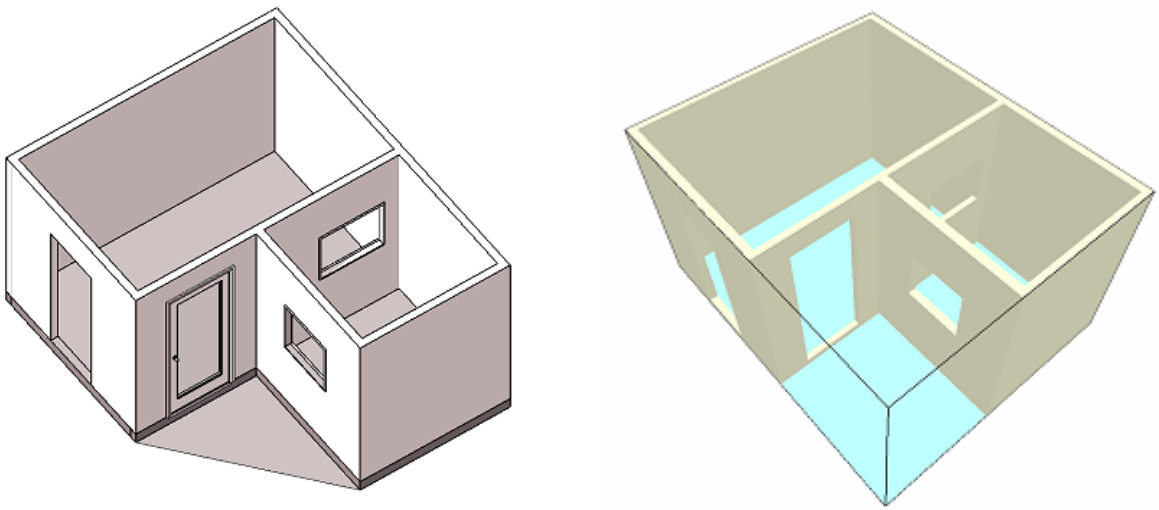


Figure 65: Floor plan of the Two-Room Model E



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 66: 3D view of the Two-Room Model E

A2.7 Two-Room Model J

This is a two-room model with two external doors having different swing directions, an external window and an internal window. A furniture item in the form of a box is also included in the smaller room (Figure 67 and Figure 68). This model has been used as a case example throughout the report.

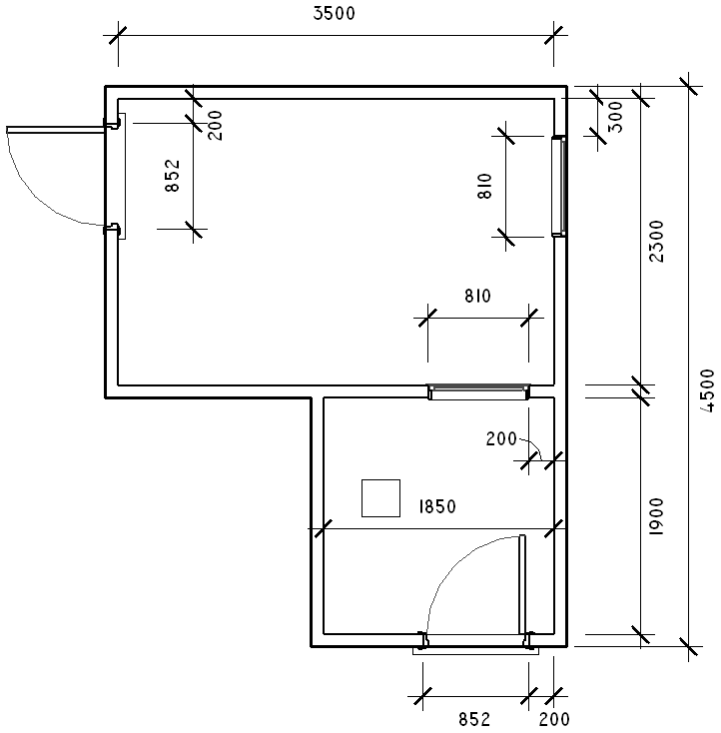
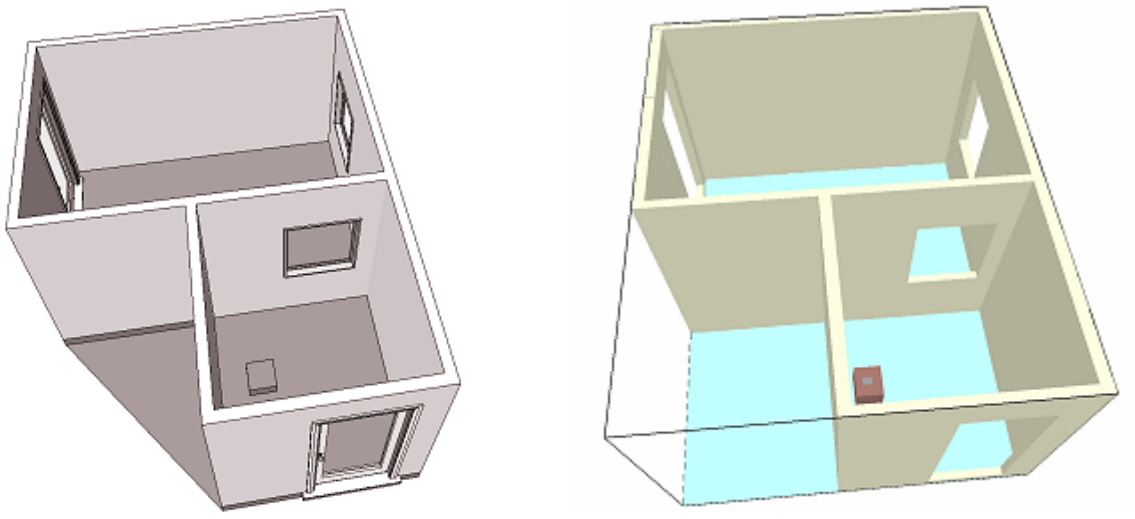


Figure 67: Floor plan of the Two-Room Model J



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 68: 3D view of the Two-Room Model J

The STEP file as well as the XML document output from the ifcSTEP Parser for this test case model is given in the Appendices.

A2.8 Multi-Room Model

This is a multi-room model with one external door, a series of internal doors and a series of windows on external walls. A box object is placed in the centre of each room representing a piece of furniture or a group of furniture items (Figure 69 and Figure 70).

This would be the most complex test case model in the project that has been correctly mapped to FDS.

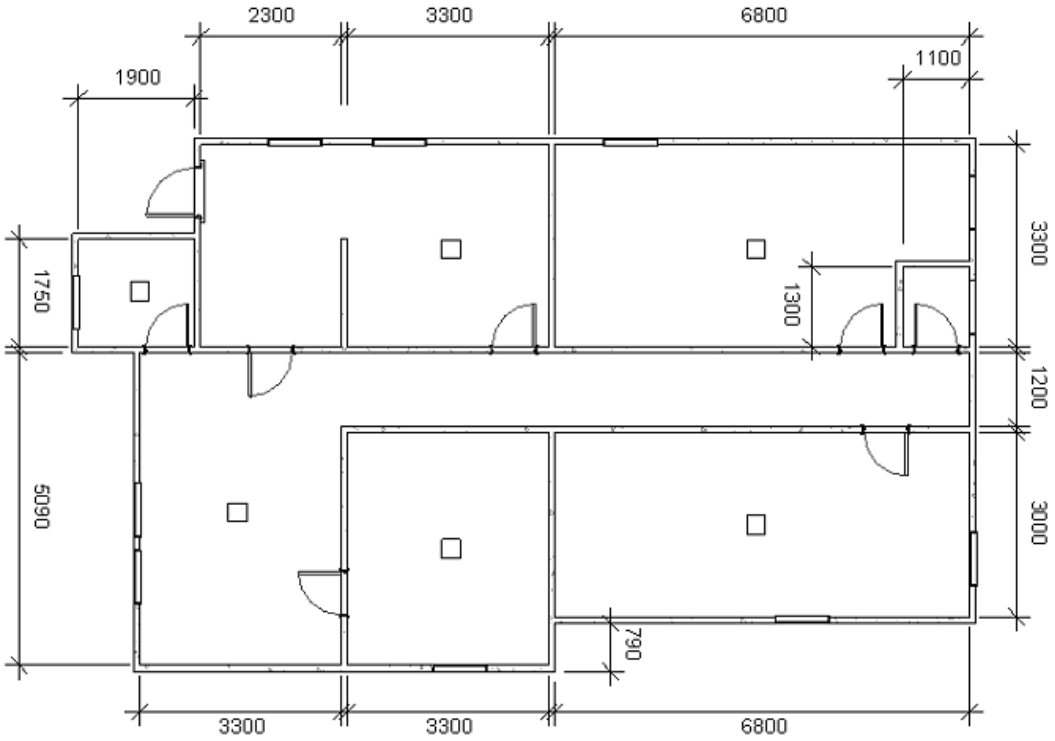
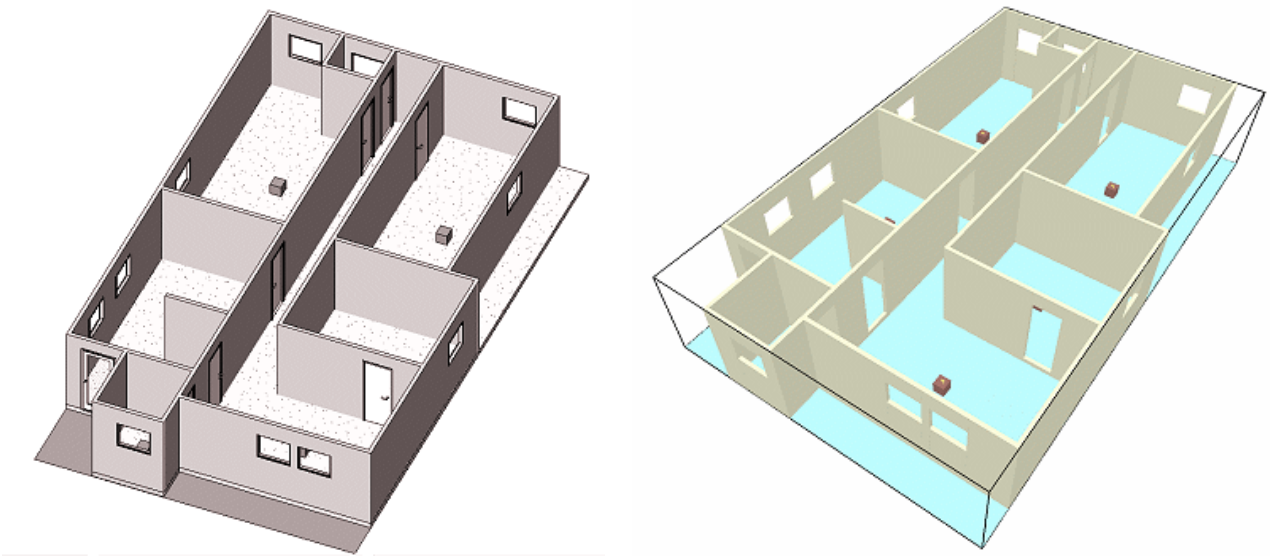


Figure 69: Floor plan of the Multi-Room Model



(a) Revit 3D view

(b) FDS output in Smokeview

Figure 70: 3D view of the Multi-Room Model

Appendix 3: STEP File of Two-Room Model J

This is an extract of the STEP file for the Two-Room Model J test case model. The original file contains a total of 9320 lines of data.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('IFC2X_PLATFORM'),'2;1');
FILE_NAME('TWOROOM2J-REVIT9','2006-09-13T11:31:06',(''),(''),'Autodesk Revit Building 9 - 1.0','20060505_2300 (Solibri IFC Optimizer)','');
FILE_SCHEMA(('IFC2X2_FINAL'));
ENDSEC;
DATA;
#1=IFCOWNERHISTORY(#6610,#1850,$,.NOCHANGE.,$,$,$,0);
#2=IFCDIRECTION((0.0,0.0,1.0));
#3=IFCGEOMETRICREPRESENTATIONCONTEXT($,$,3,1.0E-6,#18,$);
#4=IFCLOCALPLACEMENT(#1501,#18);
#5=IFCCARTESIANPOINT((0.0,0.0));
#6=IFCDIRECTION((1.0,0.0,0.0));
#7=IFCWALLSTANDARDCASE('2K3MuIuD97deqAbRvz$6hO',#1,'Basic Wall:90293',$,'Basic Wall:69872',#1671,#6624,'90293');
#8=IFCWALL('3bJsw0iyTEvAnNgH_nlLK6',#1,'Basic Wall:97100',$,'Basic Wall:69872',#1672,#6620,'97100');
#9=IFCWALLSTANDARDCASE('2K3MuIuD97deqAbRvz$6ho',#1,'Basic Wall:90271',$,'Basic Wall:69872',#1673,#6623,'90271');
#10=IFCWALL('2HpktqfiHC3h3Pak_nHQMP',#1,'Basic Wall:92178',$,'Basic Wall:69872',#1668,#6618,'92178');
#11=IFCWALLSTANDARDCASE('2HpktqfiHC3h3Pak_nHQMB',#1,'Basic Wall:92160',$,'Basic Wall:69872',#6600,#6622,'92160');
#12=IFCWALLSTANDARDCASE('2K3MuIuD97deqAbRvz$6hb',#1,'Basic Wall:90248',$,'Basic Wall:69872',#6599,#6621,'90248');
#13=IFCPROPERTYSINGLEVALUE('Assembly Code',$,IFCLABEL(''),$);
#14=IFCPROPERTYSINGLEVALUE('Assembly Description',$,IFCLABEL(''),$);
#15=IFCWALLSTANDARDCASE('2HpktqfiHC3h3Pak_nHQMl',#1,'Basic Wall:92196',$,'Basic Wall:69872',#6596,#6619,'92196');
#16=IFCPROPERTYSINGLEVALUE('IsExternal',$,IFCBOOLEAN(.T.),$);
#17=IFCSLAB('2K3MuIuD97deqAbRvz$6ja',#1,'Floor:90377',$,'Floor',#6606,#6553,'90377',.FLOOR.);
#18=IFCAXIS2PLACEMENT3D(#56,$,$);
...
```

```
...
#1682=IFCOPENINGELEMENT('23FulpWxHCXgbJTdFNHy6D',#1,'Ext Single In-Timber-1Lite-b (NZ):800 x  2032:104685:1',
        $,'Opening',#1670,#6550,$);
#1683=IFCOPENINGELEMENT('2fKFkJOYrDIAaWH3Bz30kp',#1,'Ext Single Out-Aluminum-Grooved (NZ):800 x 1980:105542:1',
        $,'Opening',#1669,#6551,$);
#1684=IFCOPENINGELEMENT('0G2DavsDrBkP3tRst2mRvs',#1,'FL 1 Aluminum-Grooved (NZ):1000 x 800:106920:1',
        $,'Opening',#1667,#6552,$);
#1685=IFCOPENINGELEMENT('1vWlxPMQH7SfbL2bilrUrt',#1,'FL 1 Aluminum-Grooved (NZ):1000 x 800:106949:1',
        $,'Opening',#1674,#6558,$);
...
```

```
...
#9304=IFCRELVOIDSELEMENT('3yTjORuJv9COIXQn2E7QtE',#1,$,$,#9,#1684);
#9305=IFCRELVOIDSELEMENT('2t6rYJVdrFXucgLIWSQfEe',#1,$,$,#10,#1683);
#9306=IFCRELVOIDSELEMENT('36VkcxEqr33BtbPO49uA6X',#1,$,$,#8,#1682);
#9307=IFCRELVOIDSELEMENT('0Hm$locTTDrwTjcBK8WVII',#1,$,$,#7,#1685);
#9308=IFCRELAGGREGATES('2GlF5w4TL7ze96B9gO1dEr',#1,$,$,#1665,(#1666,#6574));
#9309=IFCRELDEFINESBYTYPE('0Oq2ID9tn2VPc7MlH6LGS2',#1,$,$,(#22,#21),#8749);
#9310=IFCPOLYLINE((#1083,#1744,#1737,#1713,#1723,#1083,#1083));
#9311=IFCRELCONTAINEDINSPATIALSTRUCTURE('1fnc6kddfExA4UR9DCOHmj',#1,$,$,(#12,#9,#7,#17,#11,
        #10,#15,#8,#20,#19,#22,#21,#1504),#1666);
ENDSEC;
END-ISO-10303-21;
```

Appendix 4: XML Document Output for Two Room Model J

```
<STEPParser-data-construction version="0.2" author="Michael Spearpoint" organisation="University of Canterbury" created="14:03:38 on Friday 15 September 2006">
 <project id="#8716" name="TWOROOM2J-REVIT9">
  <units id="#8748" length="MILLIMETRE" />
 </project>
 <storey id="#1666" name="Level 1">
  <wall id="#12" name="Basic Wall:90248" length_x="3600" length_y="0" height="2550" pos_x="6297" pos_y="9691" pos_z="0">
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <wall id="#9" name="Basic Wall:90271" length_x="0" length_y="-2500" height="2550" pos_x="9947" pos_y="9741" pos_z="0">
   <window id="#22" name="FL 1 Aluminum-Grooved (NZ):1000 x 800:106920" width_x="0" width_y="-810" height="1010" sill="900" soffit="1910" pos_x="9947" pos_y="9341" pos_z="900" />
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <wall id="#7" name="Basic Wall:90293" length_x="-3600" length_y="0" height="2550" pos_x="9897" pos_y="7291" pos_z="0">
   <window id="#21" name="FL 1 Aluminum-Grooved (NZ):1000 x 800:106949" width_x="-810" width_y="0" height="1010" sill="900" soffit="1910" pos_x="9697" pos_y="7291" pos_z="900" />
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <slab id="#17" name="" type="UNDEFINED" pos_x="6297" pos_y="5241" pos_z="0">
   <materials id="#6542">
    <material id="#6539" description="$" thickness="150" />
   </materials>
  </slab>
  <wall id="#11" name="Basic Wall:92160" length_x="0" length_y="-2000" height="2550" pos_x="9947" pos_y="7241" pos_z="0">
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <wall id="#10" name="Basic Wall:92178" length_x="-1950" length_y="0" height="2550" pos_x="9897" pos_y="5291" pos_z="0">
   <door id="#19" name="Ext Single Out-Aluminum-Grooved (NZ):800 x 1980:105542" width_x="-800" width_y="0" height="1980" soffit="2130" pos_x="9697" pos_y="5291" pos_z="150" />
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <wall id="#15" name="Basic Wall:92196" length_x="0" length_y="1900" height="2550" pos_x="7997" pos_y="5341" pos_z="0">
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <wall id="#8" name="Basic Wall:97100" length_x="0" length_y="-2300" height="2550" pos_x="6347" pos_y="9641" pos_z="0">
   <door id="#20" name="Ext Single In-Timber-1Lite-b (NZ):800 x 2032:104685" width_x="0" width_y="-852" height="2030" soffit="2180" pos_x="6347" pos_y="9467" pos_z="150" />
   <materials id="#6541">
    <material id="#6540" description="Default Wall" thickness="100" />
   </materials>
  </wall>
  <furniture id="#1504" name="SimpleBox:SimpleBox:113595" pos_x="8356" pos_y="6279" pos_z="148" />
 </storey>
 <storey id="#6574" name="Level 2" />
</STEPParser-data-construction>
```

Appendix 6: Snippets of ifcSTEP-FDS source codes

A6.1 XML Parsing Setup Procedure

```
if (! ($xmlparser = xml_parser_create())) {
    die ("Cannot create parser");
}

// Set element handler by calling start_tag and end_tag functions
xml_set_element_handler($xmlparser, "start_tag", "end_tag");

if ($tmp_name and $errormessage == null) {
    $filename = $tmp_name;
    if (!($fp = fopen($filename, "r"))) {
        die ("Cannot open " . $filename);
    }
    // read file and remove all white spaces with regex
    while ($data = fread($fp, 4096)) {
        $data = eregi_replace(">" . "[[:space:]]+" . "<", "><", $data);
        if (!xml_parse($xmlparser, $data, feof($fp))) {
            $reason = xml_error_string(xml_get_error_code($xmlparser));
            $reason .= xml_get_current_line_number($xmlparser);
            die ($reason);
        }
    }
    xml_parser_free($xmlparser);
}
```

A6.2 Attributes Retrieval Function

```
function start_tag($parser, $name, $attribs)
// $parser = handle to the parser
// $name = name of the current tag
// $attrib = an array containing any attributes of the current tag
{
    // Validate ifcSTEP Parser specific data header
    if ($name == "STEPPARSER-DATA-CONSTRUCTION") {
        $ifcstepparsercompliant = 'OK';
        if (is_array($attribs)) {
            // show attributes
            while (list($key, $val) = each($attribs)) {
                if ($key == "AUTHOR") {
                    $xmldocauthor = $val;
                }
                if ($key == "CREATED") {
                    $xmldoccreateddate = $val;
                }
            }
        }
    } else {
        // Other elements & attributes
        if (is_array($attribs)) {
            // Loop through and extract attributes
            while (list($key, $val) = each($attribs)) {
                // get header information
                getHeaderInfo("PROJECT", "NAME");
                // get units
                getLengthUnits("UNIT", "LENGTH");
                // get walls data
                getWallID("ID");
                getWallPos_X("POS_X");
                getWallPos_Y("POS_Y");
                getWallPos_Z("POS_Z");
                getWallLength_X("LENGTH_X");
                getWallLength_Y("LENGTH_Y");
                getWallHeight("HEIGHT");
                // get wall openings
                ...
                // get furniture position
                ...
            }
        }
    }
}
```

A6.3 Graphics Generation Algorithm

```
$imWidth = 308;
$imHeight = $imWidth*(($ybar1-$ybar0)/($xbar1-$xbar0));

// Scale image to fit the graphics area
if ($xbar1 - $xbar0 > 0) {
    $xscale = 0.98 * $imWidth / ($xbar1 - $xbar0);
    $yscale = 0.98 * $imHeight / ($ybar1 - $ybar0);
} else {
    $xscale = 0;
    $yscale = 0;
}

ImageFill($myImage, 0, 0, $white);
Imageinterlace($myImage, 1);
// Draw walls as filled rectangles with color black
for ($counter = 1;$counter <= $numberofwalls;$counter++) {
    // subtract $xbar0 and $ybar0 to translate it to the origin
    // subtract y-ordinates from $ybar to mirror the image
    $tx1 = $xscale * ($x1[$counter] - $xbar0);
    $tx2 = $xscale * ($x2[$counter] - $xbar0);
    $ty1 = $yscale * ($ybar1 - $y2[$counter]);
    $ty2 = $yscale * ($ybar1 - $y1[$counter]);
    Imagefilledrectangle($myImage, $tx1, $ty1, $tx2, $ty2, $darkgrey);
}
// Show wall opening as filled rectangles with color white
for ($counter = 1;$counter <= $numberofopenings;$counter++) {
    $tx1 = $xscale * ($hx1[$counter] - $xbar0);
    $tx2 = $xscale * ($hx2[$counter] - $xbar0);
    $ty1 = $yscale * ($ybar1 - $hy2[$counter]);
    $ty2 = $yscale * ($ybar1 - $hy1[$counter]);
    Imagefilledrectangle($myImage, $tx1, $ty1, $tx2, $ty2, $white);
}
Imagestring($myImage, 2, 18, $imHeight, ("Building Plan, created "
. $timestamp), $blk);
imagegif($myImage, $image_path);
Imagedestroy($myImage);
```

A6.4 Wall Mapping Algorithm

```
// MAPPING WALLS - all in original unit of measurements
$counter = 0;
$ntemp = 0;

for ($counter = 1; $counter <= $numberofwalls; $counter += 1) {
    $wall_offset[$counter] = $wall_thickness[$counter] / 2;
    // Compute sextuplet coordinates
    if ($wall_length_y[$counter] == 0) { // wall is parallel the x-axis
        $x1[$counter] = $wall_x[$counter];
        $y1[$counter] = $wall_y[$counter] - $wall_offset[$counter];
        $z1[$counter] = $wall_z[$counter];
        $x2[$counter] = $wall_x[$counter] + $wall_length_x[$counter];
        $y2[$counter] = $wall_y[$counter] + $wall_offset[$counter];
        $z2[$counter] = $wall_z[$counter] + $wall_height[$counter];
    } Else { // if ($wall_length_x[$counter] = '0') or wall is parallel the y-axis
        $x1[$counter] = $wall_x[$counter] - $wall_offset[$counter];
        $y1[$counter] = $wall_y[$counter];
        $z1[$counter] = $wall_z[$counter];
        $x2[$counter] = $wall_x[$counter] + $wall_offset[$counter];
        $y2[$counter] = $wall_y[$counter] + $wall_length_y[$counter];
        $z2[$counter] = $wall_z[$counter] + $wall_height[$counter];
    }
}
```

A6.5 Wall Opening Mapping Algorithm

```
// ============================================================================
// MAPPING WALL OPENINGS - all in original unit of measurements
$counter = 0;
$ntemp = 0;

for ($counter = 1; $counter <= $numberofopenings; $counter += 1) {
    // Offset a further 25% of wall thickness on each side to ensure hole is deeper than the wall thickness
    $wall_offset[$counter] = ($wall_thickness[$counter] / 2) * 1.5;
    // Compute sextuplet coordinates
    if ($opening_width_y[$counter] == 0) { // opening is on wall that is parallel the x-axis
        $hx1[$counter] = $opening_x[$counter];
        $hy1[$counter] = $opening_y[$counter] - $wall_offset[$counter];
        $hz1[$counter] = $opening_z[$counter];
        $hx2[$counter] = $opening_x[$counter] + $opening_width_x[$counter];
        $hy2[$counter] = $opening_y[$counter] + $wall_offset[$counter];
        $hz2[$counter] = $opening_z[$counter] + $opening_height[$counter];
    } Else { // if ($opening_width_x[$counter] = '0') or parallel the y-axis
        $hx1[$counter] = $opening_x[$counter] - $wall_offset[$counter];
        $hy1[$counter] = $opening_y[$counter];
        $hz1[$counter] = $opening_z[$counter];
        $hx2[$counter] = $opening_x[$counter] + $wall_offset[$counter];
        $hy2[$counter] = $opening_y[$counter] + $opening_width_y[$counter];
        $hz2[$counter] = $opening_z[$counter] + $opening_height[$counter];
    }
}
```

A6.6 Furniture Item Mapping Algorithm

```
// ============================================================================
// MAPPING FURNITURE - all in original unit of measurements
$counter = 0;
$ntemp = 0;
// A 150mm x 150mm x 300mm high box representing the furniture entity
for ($counter = 1; $counter <= $numberoffurnitures; $counter += 1) {
    $fx1[$counter] = $furniture_x[$counter] - (150 * $Length_factor);
    $fy1[$counter] = $furniture_y[$counter] - (150 * $Length_factor);
    $fz1[$counter] = $furniture_z[$counter];
    $fx2[$counter] = $furniture_x[$counter] + (150 * $Length_factor);
    $fy2[$counter] = $furniture_y[$counter] + (150 * $Length_factor);
    $fz2[$counter] = $furniture_z[$counter] + (300 * $Length_factor);
}
```

Appendix 7: Relevant IFC & BIM Resources on the Internet

Article on Building Information Modelling.	Wikipedia http://en.wikipedia.org/wiki/Building_Information_Modeling
Article on Building Information Models.	(WBDG) Whole Building Design Guide. http://www.wbdg.org/design/bim.php
Articles on US National BIM Standards.	US National Institute of Building Sciences. http://www.nibs.org/
Autodesk Revit Building, software information and download site.	Autodesk http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=7142518
CIS/2, IFC, VRML, and X3D software information and download site.	NIST Building and Fire Research Laboratory http://cic.nist.gov/vrml/cis2.html
DDS Building Services Partner software information and download site.	Data Design System http://www.dds-bsp.co.uk/Products.html
IFC Quick Browser, software information and download site.	G.E.M. Team Solutions http://www.team-solutions.de/?page_id=19
IFC support add-ons information and download site.	Graphisoft http://www.graphisoft.com/support/ifc/downloads/public/
IfcStoreyView and other software utilities information and download site.	Forschungszentrum Karlsruhe Institute for Applied Computer Science http://www.iai.fzk.de/english/projekte/VR-Systems/html/Download/Software/index.html
Octaga Modeller for IFC and other software utilities information and download site.	Octaga Modeller http://www.octaga.com/download_modeller.html
SketchUp 3D software information and download site.	SketchUp from Google http://www.sketchup.com/?section=products
Software and utilities information and download site.	BIM Resources @Georgia Tech http://bim.arch.gatech.edu/app/bimtools/tools_list.asp
Solibri IFC Optimizer, software information and download site.	Solibri http://www.solibri.com/index.php?option=com_content&task=view&id=23&Itemid=32