10th International Congress on Advances in Civil Engineering, 17-19 October 2012
Middle East Technical University, Ankara, Turkey

Representation of Topological Building Information for Creating Shortest Evacuation Paths during Building Emergencies

G. Guven[1], M. Ayhan[2], E. Ergen[3], T. Sonmez[4], O. Kurc[5], M.A. Erberik[6], M.T. Birgonul[7]

[1]Department of Civil Engineering, Istanbul Technical University, Istanbul, Turkey, gursans.guven@itu.edu.tr
[2]Department of Civil Engineering, Gazi University, Ankara, Turkey, muratayhan@gazi.edu.tr
[3]Department of Civil Engineering, Istanbul Technical University, Istanbul, Turkey, esin.ergen@itu.edu.tr
[4]Departmentof Computer Engineering, Istanbul Technical University, Istanbul, Turkey, sonmeztu@itu.edu.tr
[5]Department of Civil Engineering, Middle East Technical University, Ankara, Turkey, kurc@metu.edu.tr
[6]Department of Civil Engineering, Middle East Technical University, Ankara, Turkey, altug@metu.edu.tr
[7]Department of Civil Engineering, Middle East Technical University, Ankara, Turkey, birgonul@metu.edu.tr

Abstract

In case of a disaster or an emergency (e.g., earthquake) in a facility, fast evacuation of the occupants is crucial. But the complex indoor environments of buildings (e.g., tall and large commercial buildings) negatively affect evacuation. Moreover, buildings are often threatened by multi-hazard situations, such as post earthquake induced hazards, enforcing the responders and occupants to deal with more than one emergency at a time. To perform effective evacuation, response teams require some information about the damaged building (e.g., building status and contents). In the current practice it is time-consuming to access this information since it is collected at the disaster area. Also, the collected information is incomplete, unstructured and inadequate for an effective indoor navigation. Thus, the required building information is needed to be represented and stored and made available for indoor navigation during emergency. The study explained in this paper proposes an approach which utilizes Building Information Model (BIM) combined with graph theory principles and sensor information to create evacuation paths for the use of responders and occupants during emergency. Industry Foundation Classes (IFC) based building models are used and the graph theory elements (e.g., nodes, edges) are defined to generate a graph network model that represents the topological map of the building. This model is integrated with a shortest path algorithm to create safe and short evacuation paths. This paper specifically provides the details on representation of graph theory elements using the building-related information obtained in IFC format, and explains the method used for creating shortest paths.

Keywords: *building information model, BIM, IFC, graph theory, shortest path algorithm, emergency response*

1 Introduction

In case of a disaster and/or an emergency situation in a building, first responders try to gather navigation information and current and potential blockages in the building to perform an effective evacuation and response operation. In the meantime, the occupants in the building require such information to leave the building safely by using the least hazardous route. When an emergency situation occurs, the occupants usually tend to panic and evacuation could not be performed uneventfully. Instead, with the panic and due to the lack of information related to the building status; occupants are usually faced with further injuries. Moreover, it is commonly observed that the disaster/emergency can trigger secondary hazards, for example fire is the mostly observed secondary disaster in the aftermath of earthquakes. Especially in large and tall commercial buildings (e.g., malls and office buildings), that have architecturally complex indoor environments, disasters/emergencies and possible secondary disasters, can cause large number of injuries and mortalities.

Although it is crucial for the first responders to obtain up-to-date building information in a timely manner for an efficient response, in the current practice assessment of the damaged building is performed by visual

investigation and by querying the people who left the building. Such a method for collecting information related to the building and the occupants is prone to errors, and it wastes time. Thus, the lack of such necessary information affects the response operations negatively, and results in increased number of mortalities (Kwan and Lee 2005, Ergen and Seyis 2008). These issues point out that there is a need for an approach to represent and store the required building information (e.g., blockages in the building, its hazardous contents) and to utilize this information during emergency for indoor navigation for occupants and responders. The study explained in this paper proposes an approach that builds on Building Information Model (BIM) combined with graph theory principles and sensor information to create shortest evacuation paths that can be used by both the responders and the occupants during emergency. Industry Foundation Classes (IFC) based building models are used to provide building-related information (e.g., geometric information of building components) and the graph theory elements (e.g., nodes, edges) are created in Java. This model is integrated with a shortest path algorithm to create safe and short evacuation paths. This paper specifically provides the details on representation of graph theory elements for creating topological mapping of buildings, and explains the method used for creating shortest paths.

2 Background Research

An effective indoor navigation approach can solve the issues observed during building emergencies and evacuation phase. This necessitates representing an up-to-date building-related information (i.e., geometry and semantics) in a format that can be used to perform indoor navigation to guide the occupants during evacuation and to lead the responders effectively inside the building. There are studies in the literature that focused on indoor navigation during emergency response, for example to create alternative evacuation paths after a disaster (Meijers et al., 2005; Pu and Zlatanova, 2005; Lee, 2007; Lee and Zlatanova, 2008; Park and Lee, 2008; Yuan and Zizhang, 2008). The previous indoor navigation studies followed different ways of obtaining the building information. While some studies retrieved building-related data from BIM (Rueppel and Stuebbe, 2008; Lee et al., 2008; 2010); others used three dimensional (i.e., 3D) Geographic Information System (GIS) models (Kwan and Lee, 2005; Lee, 2005; 2007; Lee and Zlatanova, 2008), BIM integrated with GIS models (Yuan and Zizhang, 2008) and/or Cellular Automata models (Park and Lee, 2008). For example, Lee (2007) gained building-related information from 3D GIS to generate graph networks for use during emergency response for navigation (Lee, 2007). Yuan and Zizhang (2008) integrated BIM with 3D GIS to obtain up-to-date geometry and semantics of buildings, while highlighting the additional need for information about threats and accessibility of the building for a successful indoor navigation.

The purpose of indoor navigation is to guide the people inside a building to safer places at safest and shortest routes to avoid damaging effects of the disaster/emergency and secondary disasters. To achieve this objective, shortest path algorithms (SPAs) are used. SPAs can calculate the possible shortest way between two points on a graph network which are composed of points (i.e., nodes) and lines (i.e., edges) that connect consecutive points. In SPAs, each edge is given a value called edge cost, such as distance, and the shortest paths are calculated based on those edge costs between nodes (i.e., distances in between nodes). The resulting minimum cost between an origin and a target node then becomes the shortest path (Ayhan et al. 2012). Thus, evacuation paths to be used during evacuation are the shortest possible paths between an origin and a target point in distance. One of the most commonly used SPA, is the Dijkstra's algorithm (Meijers et al., 2005; Lee, 2007; Lee and Zlatanova, 2008). For example, Meijers et al. (2005) used a semantic model of a building which stored building geometry and other important building information in a graph network and utilized Dijkstra's SPA for 3D indoor emergency evacuation. Also, the relationship of graph network components (i.e., node and edge) and computation details were provided in some of the previous studies (Lee, 2007; Lee and Zlatanova, 2008). The Dijkstra's algorithm was used in the study explained in this paper.

Some of the previous indoor navigation related studies obtained the required building information (i.e., building geometry and semantics) in IFC format. For example, Lee et al. (2008; 2010) used IFCs for computing accessible distances based on a length-weighted graph structure taking handicapped people in wheelchairs into consideration. These studies identified the IFC classes that are required for creating a graph network based on the building geometry and semantics (e.g., *IfcSpace*, *IfcDoor*, *IfcStair*, *IfcRelSpaceBoundary*). Although the IFC standard gives all the building-related information, such as the building elements and their relationships, this is not sufficient for creating graph networks. The graph network elements (i.e., nodes and edges), are needed to create topological mapping of a building and these classes are not represented in IFCs. Thus, the graph network elements are additionally created for the calculation of evacuation paths. The study explained in this paper followed this approach. Previous studies that used IFCs for representing building information highlighted the importance of *IfcSpace* class since it provides the basic information about a space (e.g., room) in a building and the relating elements (e.g., doors, walls, columns). This information can be used to establish the topological

connection in between the spaces, for example through doors (Lee et al., 2008). Additionally, there are studies that utilized the building information in IFC format to create indoor graph networks for different purposes, such as indoor circulation code checking (Eastman et al., 2009) and creating fire simulations (Spearpoint, 2007). The need for and the importance of *IfcSpace* class was emphasized in these studies as well, in addition to other classes such as *IfcWall*, *IfcDoor*, *IfcStair*, *IfcRelSpaceBoundary*. Another study that utilized IFC data (Taneja et al., 2011), developed a representation of a navigation network based on the Construction Operations Building information exchange (COBie) standard, which contains specific building information retrieved from IFC files. Taneja et al.'s (2011) representation contains mainly four items (i.e., node, coordinate, edge and node-tree) to create the proposed navigation network. On the other hand, in the study explained in this paper; additional classes were used such as node, space, building storey, edge along with a class to read the related IFC data, a class to manage the sensor data and the results of the vulnerability assessment, and finally the uppermost graph network class. Also, unlike the overall previous IFC-based studies, this study integrates the blockage information obtained from sensors and the vulnerability assessment results with the graph network generation process. In this study, the building information in IFC format is extracted from BIM and included in a model for developing shortest evacuation paths for use during emergencies in buildings. Using Java programming language, graph theory elements (e.g., nodes, edges) are defined to create a customized graph network model of the building, and this model is integrated with Dijkstra's algorithm to create safe and short evacuation paths.

3 Proposed System Framework

The study explained in this paper is a part of an ongoing project in which a framework that utilizes BIM integrated with sensors and a vulnerability assessment algorithm to provide the blockage and vulnerability information of a facility that is under the threat of multi-hazard situations. The main purpose is to present guidance to occupants during evacuation and to assist the first responders in rescue operations by determining the blockage and vulnerable locations. The proposed system framework composes of three steps (Figure 1). Creating a customized graph network model by defining graph network elements (i.e., nodes and edges) using the information stored in an IFC-based building information model is the first step for developing the evacuation paths. The elements of the customized graph network model will be generated according to predefined structural rules, details of which are given in Ergin et al. (2012). This is referred to as customization in this study. For example, the area that is bounded by four columns, four structural and/or non-structural walls, a slab, a ceiling or a suspended-ceiling, and other non-structural units that might cause a blockage (e.g., cupboards) make up a space (e.g., corridor) that is represented by a node. Each node in the graph network will be connected to other nodes via edges. In addition, there will be room nodes for rooms in the building, and door nodes at door frames. For the second step, different types of sensors that are deployed inside the building will provide blockage status about the building. The status information retrieved from the sensors and the results of the vulnerability assessment integrated with the customized graph network model that is driven from the IFC-based model constitutes a deformed graph network model that indicates the vulnerable and blocked locations. Finally, the shortest evacuation paths are calculated based on the up-to-date building conditions. More information can be found in Ergin et al. (2012) and Guven et al. (2012). This study explained in this paper focuses on the first step that deals with the generation of the customized graph network model.

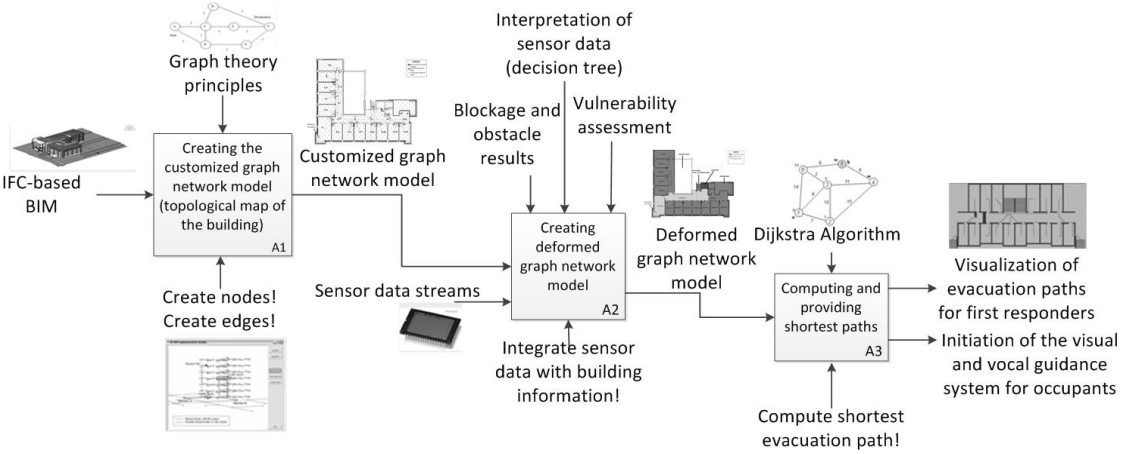


Figure 1. Proposed system framework.

4 Implementation Details

A software model is being developed to apply the proposed framework. The primary input of this software model is the BIM file that is originally created in a BIM tool (e.g., Autodesk Revit, Graphisoft Archicad) and exported to IFC format. To detect the required IFC structures in the exported file, the IFC libraries of an open source IFC reader/writer named Open IFC Java Toolbox (Tulke et al., 2010) was used. Using Java language during programming ensures that the developed prototype can run on any computers, netbooks and tablets that have Java Virtual Machine (JVM) under Windows or Linux operating systems. The proposed system first detects the related IFC structures and stores them. The software model that is being developed is designed to have six layers, which are given as follows:

1) Manager layer: This is the top layer of the developed software model which manages all the software processes.
2) File manager layer: The content of a file exported to IFC format can have a relatively complex structure due to the massive amount of information that can be stored within the context of the IFC standard. For example, even if no value is entered for a specific data type in IFCs (e.g., manufacturer information of a window) during design, that data type will still exist in the exported IFC file as "null", "$", or "-1"; indicating that no value is entered for that data type. On the other hand, there is also some information stored in the IFC file that is not used in the proposed framework. Such null and/or unnecessary data should be eliminated while processing the IFC file. Thus, the IFC file should be processed to identify the required data stored in the file to send the data to the related data structures. The second layer of the software model is created to perform this task by reading the IFC file to extract the necessary data stored in IFCs and sending the data to related data structures.
3) Topology constructor layer: This layer is responsible for running the recursive algorithm for *IfcBuildingStorey* structures to create the topological map of the objects that are physically connected to each other.
4) Sensor layer: This layer integrates the information obtained from the sensors and the results of the vulnerability assessment in the topological map created in the previous layer. This layer is responsible for increasing or decreasing the distances between two nodes in the topological map according to the results obtained from the sensors and the vulnerability assessment. If the results indicate that a node is blocked, then the node's distances with its neighboring nodes are increased to ensure that the blocked node is not included in the SPA calculations.
5) Search engine layer: This layer runs the SPA (i.e., Dijkstra) on the topological map of the building.
6) Visualization layer: This is the layer responsible for generating the interface and visualizing the created Dijkstra trees.

The following paragraphs highlight the important details on the development procedures of the proposed software model:

- The importance of *IfcSpace* structure was emphasized by the previous studies in the literature. The proposed framework also uses the *IfcSpace* structure for obtaining the necessary information related to the building. When designing the building model, the building is thought to be constituting of subdivisions which will be the basis for calculating the shortest and safest evacuation paths. The subdivisions in a building model are called the "spaces" in the building (e.g., a room, a part of a room, a corridor). The areas in the BIM model such as rooms, halls, corridors, are represented with *IfcSpace* in the IFC standard. In IFC, an *IfcSpace* object composes of building elements such as doors (i.e., *IfcDoor*), columns (i.e., *IfcColumn*), windows (i.e., *IfcWindow*), slabs (i.e., IfcSlab) and walls (i.e., *IfcWall*).
- Since the proposed framework obtains the distance-related data from *IfcSpace* objects; they become the fundamental elements for determining the shortest evacuation paths. The system calculates the distance between the mid-point of an *IfcSpace* (1) to the mid-point of its neighboring *IfcSpace*, and (2) to the closest *IfcDoor*, or (3) to the closest *IfcWindow* since windows can be used as an exit at the ground floor during an emergency. The system will guide the occupants to the shortest alternative (i.e., in distance).
- The process explained previously, is repeated for all the *IfcSpace* objects in the IFC file. The geometric dimensions of *IfcSpace* objects are already available in the IFC file and this is combined with the new distance values (i.e., between neighboring *IfcSpace*, *IfcWindow*, *IfcDoor* objects) in an appropriate data structure. Consequently, an "*IfcSpace* Network" is generated from *IfcSpace* objects that are connected to each other. This network will then be processed by the SPA.
- The Dijkstra's SPA is utilized in this study. This algorithm takes one of the nodes from the network (i.e., in this case *IfcSpace* Network) as the origin (i.e., root) and calculates the distances and paths between the origin node and the other nodes within the network. As explained before, nodes in the *IfcSpace* Network are located at the mid-points of each *IfcSpace* object. Thus, the requirements for an area to be accepted as a space need to be defined (i.e., *IfcSpace*). In this study, an area that is bounded by four structural columns,

four beams, two infill walls, one suspended ceiling and other non-structural entities is assumed to constitute an *IfcSpace* object. The reason behind this approach is that it was aimed to ensure each *IfcSpace* object stays within the limits of the sensor coverage zones. More information can be found in Ergin et al. (2012).

- The principal components of Dijkstra's algorithm are "M" total number of edges and "N" total number of nodes. It is clear that in a medium-scaled building, there will be tens of *IfcSpace* objects and hundreds of edges that are connecting these objects. It is observed that as the number of *IfcSpace* objects and the number of their connections to each other increase, the system performance decreases. Thus, it will be inefficient to run the Dijkstra's algorithm and generate an *IfcSpace* Network for every single *IfcSpace* to determine the distance to other *IfcSpace* objects. It should be noted that the main goal of the SPA algorithm used in this study is to guide the occupants to the nearest *IfcSpace* that involves an exit. Based on this idea, it is more appropriate to select each node with an exit (e.g., window at the ground floor, door at every floor) as the origin (i.e., root), and run the SPA one by one only for such nodes and generate the related trees. The trees are then filtered starting from the root nodes through the leaves. This filtering process ensures that each node is contained in the tree in which it has its minimum value (i.e., distance), compared to its values in other trees. Consequently, the node stays in the tree where it has the minimum value and is removed from all other generated trees. This eliminates convergent trees and increases the system efficiency. Also, the complexity of the visualization of the shortest paths is avoided.
- In IFC, all objects are defined in a local coordinate system instead of a global coordinate system. Treldal (2008) stated that, to identify the placement of objects in relation to each other, it is required to derive the global coordinates. Similarly, to be able to create trees in Dijkstra algorithm, each object's local x-y-z coordinate values should be transposed to the building's global coordinates. The procedure for processing the local coordinate values that is described by Treldal (2008) was used. Since this transposition algorithm is very complex and processed recursively, it primarily determines the performance and the speed of the developed software model.

5 Prototype Details

The six-layered prototype has two main classes: (1) the *GraphNetwork* class and (2) the *TIfcConnect* class. The *GraphNetwork* class, manages the *IfcSpace* Network and processes the flow of all the related algorithms. The *TIfcConnect* class, on the other hand, is created in the first layer and is in charge of reading and processing the data stored in *IfcBuildingStorey* and *IfcRelSpaceBoundary* classes that are found in the IFC file. While the *IfcBuildingStorey* class stores the information related to each floor of the building model; *IfcRelSpaceBoundary* class enables accessing the data structures stored in *IfcProduct* type that is present in *IfcSpace* classes.

5.1 Software Architecture and Modeling

The first step is to retrieve the *IfcBuildingStorey* class in a list form from the IFC file. Using the "*RelatingObject*" list of each *IfcBuildingStorey* object, and then using the "*RelatedObject*" list that is present in each "*IfcRelAggregates*" member of this list; the list of *IfcSpace* can be gathered. To gather this required information, the relationship between *IfcBuilding*, *IfcBuildingStorey*, *IfcSpace* and *IfcProduct* is used (Figure 2).

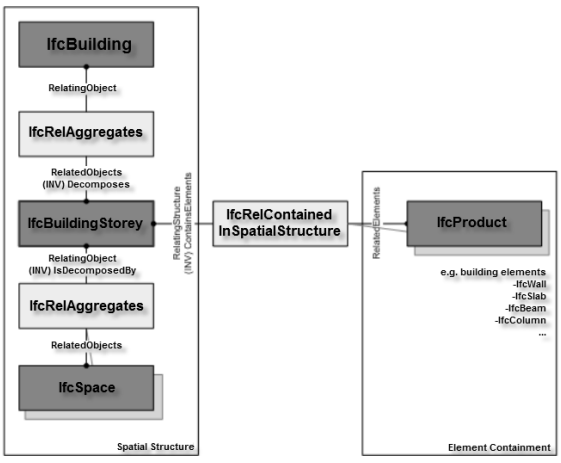


Figure 2. Relationship between *IfcBuilding-IfcBuildingStorey-IfcSpace* and *IfcProduct* (buildingSmart).

After obtaining the *IfcSpace* lists of each floor, the second step is to retrieve the *IfcProduct* objects belonging to each *IfcSpace* object. To do this, *IfcRelSpaceBoundary* class is used and the *IfcGlobalId* value of "*relatingSpace*" variable belonging to *IfcRelSpaceBoundary* class is checked whether the *IfcGlobalId* value is the same as the related *IfcSpace* object's *IfcGlobalId* value. If yes, it then becomes possible to reach to the *IfcProduct* data of the related *IfcSpace* object using the "*relatedBuildingElement*" variable of the *IfcRelSpaceBoundary* object. This data extraction process starting from the *IfcBuildingStorey* and finally ending at *IfcProduct* is constituted recursively.

There are several classes derived from *IfcProduct* class and the ones that are used in the proposed software model are:

1. *IfcWall*: This class is required while generating the visual interface.
2. *IfcDoor*: This class is necessary for determining the connections between two nodes in the network to generate the trees that will be used by Dijkstra algorithm when at least one of the nodes is on a door object.
3. *IfcWindow*: The need for this class is similar to the need for *IfcDoor* class. However, it is only needed at the ground floor of a building.
4. *IfcRamp*: This class will provide the geometry and other semantic information regarding the ramps within the building.
5. *IfcStair*: This class is needed to gather the information related to the geometry and semantics of the stairs within the building.
6. *IfcVirtualElement*: When a space is created in the BIM tool, some boundaries of the spaces might be virtual boundaries that do not really exist. In IFC, such boundaries are represented with *IfcVirtualElement*.

5.1.1 Using the IFC file for creating the customized graph network model

It is previously stated that although the IFC standard gives all the building-related information, such as the building elements and their relationships, this is not sufficient for creating graph networks. The graph network elements (i.e., nodes and edges) are needed to create the topological mapping of a building. This necessitates creation of these additional classes and calculation of the evacuation paths on the developed graph network model. The proposed software model UML class diagram is given in Figure 3.

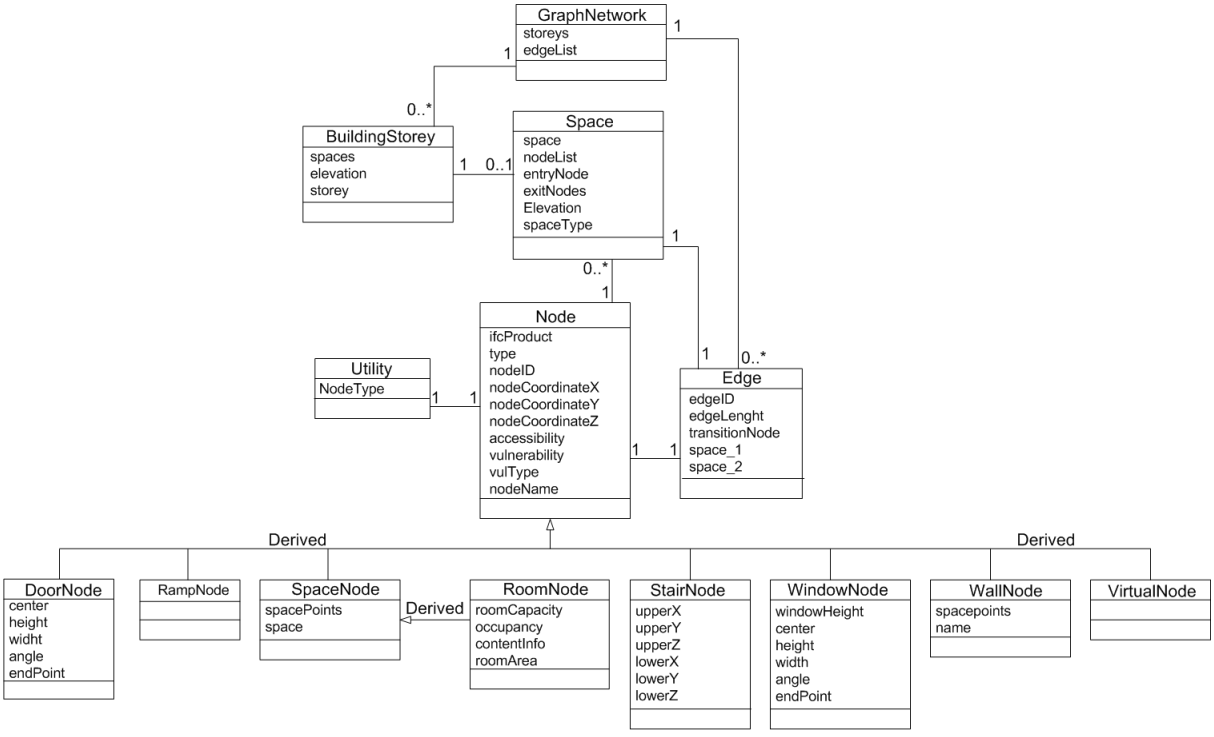


Figure 3. UML class diagram of the proposed software model.

This section of the paper provides the details about the created classes and their properties. The generated classes can be grouped into seven, namely: (1) *Node* and its derivatives, (2) *Space*, (3) *BuildingStorey*, (4) *Edge*, (5) *TIfcConnect*, (6) *TSensorConnect*, and (7) *GraphNetwork*. These classes are sequential from the innermost to the outermost in the software layer and generated in accordance with object-oriented programming principles.

a. *Node:* *Node* class is created to model and represent all the nodes (i.e., points) within the Dijkstra trees. It includes attributes (i.e., variables) such as *IfcProduct*, *NodeType*, *NodeId*, *NodeCoordinateX*, *NodeCoordinateY*, *NodeCoordinateZ*, *accessibility*, *vulnerability*, and *NodeName* needed to represent the related building elements, the node's coordinates and its blockage status, and such. For example, *NodeType* is a Java "Enum" list and it indicates whether a node is *RoomNode*, *SpaceNode*, *DoorNode*, *StairNode*, *RampNode*, *WindowNode*, *WallNode*, *VirtualNode* or none. For each type, a related class is derived from the *Node* class. Thus, the *Node* class is the parent class to all its derivatives, and these classes inherit all the attributes of the *Node* class. Another important variable is the *NodeId* given that it holds the *IfcGlobalID* which is a unique key in IFC that distinguishes *Node* objects from each other. Moreover, *accessibility* and *vulnerability* variables state whether the node is blocked or not, in accordance with the vulnerability assessment results and the information obtained from the sensors.

i. *RampNode*: Ramps are one of the building elements that are needed to be represented in the graph network as they are a means of connection between two building floors. Thus, *RampNode* class represents the nodes on the ramp structures (i.e., inclined transition parts of buildings designed for use by occupants with wheelchairs) in the building.
ii. *SpaceNode*: This class represents the nodes related to *IfcSpace* objects.
iii. *RoomNode*: This class is derived from *SpaceNode* class and inherits all the attributes from its parent classes (i.e. *Node* and *SpaceNode*). In the IFC file, if an *IfcSpace* object represents a room space, the name of that object starts with the word "Room". In such cases, that *IfcSpace* object will be represented with *RoomNode*.
iv. *StairNode:* This class represents the nodes on stair elements and thus, it is in coordination with *IfcStair* class. Each stair in the building is envisioned to be represented with two *StairNode* classes, one at the uppermost and one at the lowermost ladders of the *IfcStair* object.
v. *VirtualNode:* This class is used to link two adjacent *IfcSpace* objects that are separated by *IfcVirtualElement* and to create an edge in between these two *IfcSpace* objects. An example to *IfcSpace* objects that are separated by an *IfcVirtualElement* can be seen in Figure 4. The connection between these two adjacent spaces is provided with an element that does not exist physically. In accordance with this study's assumption related to area boundaries (i.e., four structural columns, four beams, two infill walls, one suspended ceiling and other non-structural entities), it is needed to create such virtual boundaries in some cases during design.

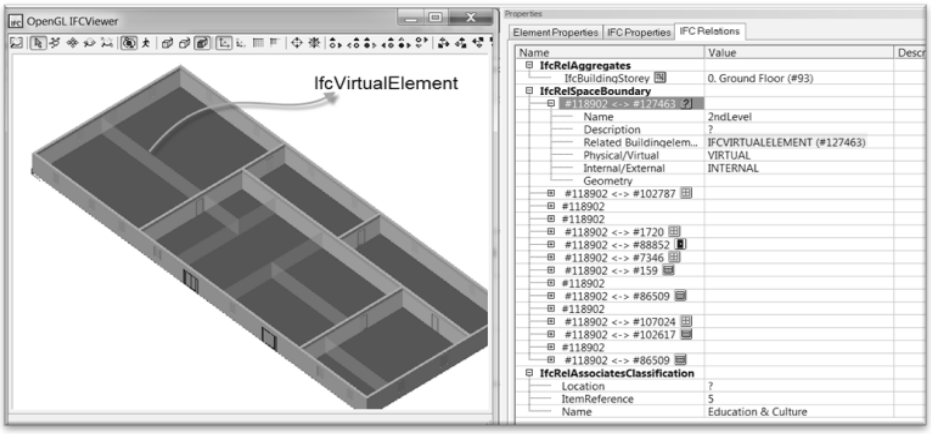
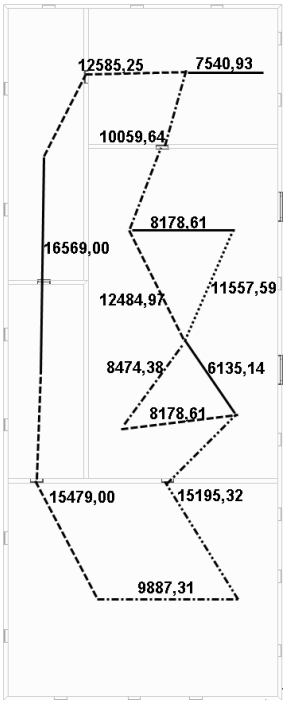


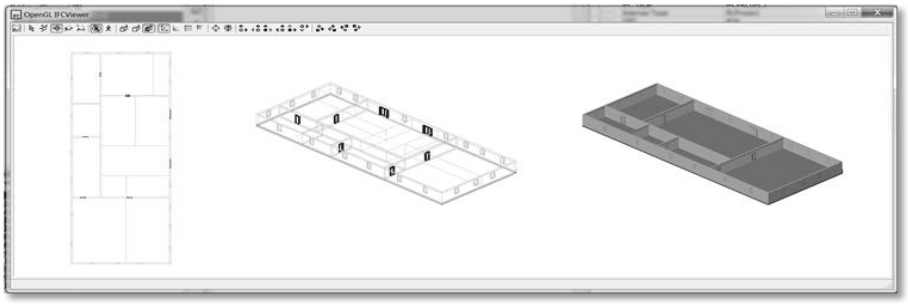
Figure 4. An example of an *IfcVirtualElement* object between two adjacent *IfcSpace* objects.

The type of boundary in between two adjacent *IfcSpace* objects also effects the calculation of the distance between *SpaceNodes*. For example, when an *IfcDoor* element is the means of connection between two *IfcSpace* objects; the inner distance of both *IfcSpace* objects to the *DoorNode* that connects them, are added. However, the distance between two *IfcSpace* objects that are connected by *IfcVirtualElement* is equal to the length of a straight line that connects each *SpaceNode*, since there are no obstacles (i.e., walls) in between. An example is illustrated in Figure 5. The lines in Figure 5a represent *Edge*s and each *Edge* is represented with different line types. *Nodes* are located at the intersections of two different type of edges. At the points where two same type of edges intersect, there are *DoorNode* objects that connect two *IfcSpace* objects. The numbers written next to the edges

indicate the length of the edge in terms of centimeters. *Edge*s between two spaces connected by *VirtualNode* objects are shown as straight lines since *IfcVirtualElement* objects connect two *IfcSpace* objects virtually since there are no physical obstacles in between these spaces (Figure 5b). In such cases, the distance is calculated by placing *VirtualNode* objects between the two *IfcSpace* objects.



(a)



(b)

Figure 5. *Edge*s generated after running the SPA on a sample IFC file, **a.** the lengths (in centimeters) and the relationships of *IfcSpace* objects with each other, and **b.** several views of the original IFC file.

vi. *WindowNode*: This class represents the nodes on the *IfcWindow* objects. Since Dijkstra trees are composed of exit locations (i.e., target points) a *WindowNode* at the ground floor can be the root for a Dijkstra tree where evacuation is possible.

vii. *DoorNode*: This class represents the nodes on the *IfcDoor* objects. Since *IfcDoor* objects can be used as exits, *DoorNode* objects can also be the root for Dijkstra trees.

viii. *WallNode*: This class represents the walls made of light materials (i.e., plaster boards) and that can be demolished in demand. For example, if gypsum is entered as the material of a wall, it is included to the list of nodes that can be used as exits. Other than that, this class will be used to gather *IfcWall* and *IfcWallStandardCase* information for creating the visual components in the visualization layer.

b. *Space*: This class represents all the nodes in the Dijkstra trees. In other words, every node in all generated Dijkstra trees corresponds to an *IfcSpace* in the IFC file, since nodes are located at the mid-points of *IfcSpace* objects. This class involves several attributes for identification and for storing the related information of each *IfcSpace* object (e.g., how many doors and windows (i.e., exits) it includes, its elevation, whether it is a corridor or a room).

c. *BuildingStorey*: This class represents floors in the building model. It holds *IfcBuildingStorey*, the first processed attribute in the software, and it generates *Space* lists by processing *IfcSpace* objects in each *IfcBuildingStorey*. It determines the relations among the spaces to generate *Edge* lists. Another duty of this class is to update the *IfcSpace* Network according to the obtained sensor and vulnerability assessment results.
d. *Edge*: There is an edge between any two adjacent nodes and they build up the *IfcSpace* Network to be used by Dijkstra trees. Thus, *Edge* class represents the relationship in between two adjacent spaces, such as the distance in between two nodes, the type of connection in between them (i.e., *DoorNode*, *WindowNode*, *VirtualNode*).
***e. TIfcConnect*:** As explained before, this class reads the IFC file, and holds *IfcRelSpaceBoundary* and *IfcBuildingStorey* lists and transmits them to the relevant layer of the software, when required.
f. *TSensorConnect*: This class manages the relations between the sensors, the vulnerability assessment results and the software model. It is responsible to transmit the data obtained from the sensors to the relevant layer and vice versa. The same is performed for obtaining the vulnerability assessment, as well. *GraphNetwork* class will run the *TSensorConnect* class in specific time intervals to achieve this data transfer.
g. *GraphNetwork*: This class is located in the top layer of the software model. It manages the whole flow of the algorithms and the *IfcSpace* Network. An object from *TIfcConnect* class is included in the *GraphNetwork* class so that it is able to read from the IFC file. Additionally, the *TSensorConnect* attribute of the *GraphNetwork* class manages the vulnerability assessment results and the data retrieved from sensors. *GraphNetwork* triggers *TIfcConnect* to start the file reading process. After having read the file, *BuildingStorey* objects are generated one by one for all members of the *IfcBuildingStorey* list that is obtained from *TIfcConnect* class. Then the "Initialize" function of the newly generated *BuildingStorey* objects is called to determine the *IfcSpace* objects of the related floor and *Space* lists are created. Following that, using the *BuildingStorey* objects and *Space* lists, as well as the *Edge* lists are generated. Thus, the *IfcSpace* Network is created.

When the "Run" function of the *GraphNetwork* class is operated, it reads the sensor data and vulnerability assessment results via *TSensorConnect* and updates the *BuildingStorey* list. Then, using the *BuildingStorey* list, it runs Dijsktra's SPA on *Space* and *Edge* objects. Following this, it transfers the generated Dijkstra trees to the visualization interface where the visual interface updates itself according to this new data. This "Run" function is a *thread* that performs independently from the main application. The "Run" function of the *GraphNetwork* is called in specific time intervals until the program or the function itself is shut.

6 Conclusions

This paper provided the details of representation of topological building information by using the information stored and extracted in IFC format. The overall framework proposes to use sensors to collect information regarding the blockage conditions inside a building, while the vulnerabilities of the building are also evaluated through a vulnerability assessment approach. The results (i.e., blockage and vulnerability conditions) are then integrated with the building information and all taken into account during the calculation of evacuation paths by shortest path algorithms. A prototype is being developed to implement the proposed framework. The primary input of this software model is the IFC based BIM file. It is aimed to create a topological map of the building by utilizing the building-related information (e.g., geometry) stored in the IFC file. Since the graph theory principles and graph network elements (i.e., nodes and edges) are not currently represented in IFCs, the required graph network elements are additionally created using Java programming language. The future work includes the completion of the prototype development process and performing related tests and identifying the potential areas for improvement.

Acknowledgements

This research is funded by a grant from the Scientific and Technological Research Council of Turkey (TUBITAK), Grant No. 109M263. TUBITAK's support is gratefully acknowledged.

References

Ayhan M., Guven, G., Ergen, E., Erberik M. A., Kurc, O., and Birgonul, M. T. (2012),"Requirements of Indoor Navigation During Emergency Response and Disaster Management", *XXXVIII IAHS World Congress on Housing*, 16-19 April, Istanbul Technical University, Istanbul, Turkey.

buildingSmart, International Alliance for Interoperability, IFC2x Edition 3 Technical Corrigendum 1. Retrieved from http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm

Eastman, C., Lee, J., Jeong, Y. and Lee, J.K. (2009). "Automatic Rule-Based Checking of Building Designs", *Automation in Construction*, Vol. 18(8), pp. 1011-1033.

Ergin, T., Baltasi, S., Erberik, M. A., Kurc, O., Ergen, E., and Birgonul, M. T. (2012). "Post-disaster sensor-based condition assessment of buildings to develop evacuation strategies", *14th International Conference of Computing in Civil and Building Engineering* (ICCBE 2012), 27-29 June, Moscow.

Ergen, E. and Seyis, S. (2008)." Utilizing Advanced Storage Technologies For Providing Local Information to Rescue Teams Following an Earthquake", *5th International Conference on Innovation in Architecture, Engineering*, June 2008, Antalya, Turkey.

Guven, G., Ergen, E., Erberik, M. A., Kurc, O., and Birgonul, M. T. (2012). "A real-time damage and vulnerability assessment of facilities for evacuation and guidance during emergency", *2012 ASCE International Conference on Computing in Civil Engineering*, 17-20 June, FL, USA.

Kwan, M. P. and Lee, J. (2005). "Emergency Response after 9/11: the potential of real-time 3D GIS for Quick emergency response in micro-spatial environments", *Computers, Environments and Urban Systems*, Vol. 29, pp. 93-113.

Lee, J. (2005). "3D GIS in Support of Disaster Management in Urban Areas", Retrieved from http://www.directionsmag.com/article.php?article_id=2049.

Lee, J. (2007). "A Three-dimensional Navigable Data Model to Support Emergency Response in Microspatial Built-environments", *Annals of the Association of American Geographers*, Vol. 97, pp. 512-529.

Lee, J.K., Eastman, C., Lee, J., Jeong, Y. and Kannala, M. (2008). "Accessible Distance Measurement Using GTMetric Graph", CIB W084, Building Comfortable and Livable Environments for all International Meeting Topic Area: Technological Tools Supporting the Accessible Environment, Georgia Institute of Technology, Atlanta, GA, USA.

Lee, J. and Zlatanova, S. (2008). "*A 3D Data Model and Topological Analyses for Emergency Response in Urban Areas*", Geospatial Information Technology for Emergency Response, 143-168, London: Taylor & Francis Group, U.K.

Lee, J.K., Eastman, C., Lee, J., Kannala, M. and Jeong, Y. (2010). "Computing Walking Distance Within Buildings Using the Universal Circulation Network", *Environment and Planning B: Planning and Design*, Vol. 37, pp. 628-645.

Meijers, M., Zlatanova, S. and Pfeifer, N. (2005). "3D Geo-Information Indoors: Structuring for Evacuation", *Proceedings of Next Generation 3D City Models*, Bonn, Germany.

Park, L. and Lee, J. (2008). "Defining 3D Spatial Neighborhoods for Topological Analyses Using a 3D Network-based Topological Data Model – CA-based Evacuation Simulation", *The International Archives of the Photogrammetry*, Remote Sensing and Spatial Information Sciences, Vol. 37(B2), pp. 305-310, Beijing.

Pu, S. and Zlatanova, S. (2005). "Evacuation Route Calculation of Inner Buildings", *Geo-Information for Disaster Management*, pp. 1143-1161, Springer Verlag, Heidelberg.

Rueppel, U. and Stuebbe, K.M. (2008). "BIM-based Indoor-emergency-navigation-system for Complex Buildings", *Tsinghua Science and Technology*, Vol. 13(S1), pp. 362-367.

Spearpoint, M. (2007). "Transfer of Architectural Data From the IFC Model to a Fire Simulation Tool", *Journal of Fire Protection Engineering*, Vol. 17(4), pp. 271-292.

Taneja, S., Akcamete, A., Akinci, B., Garrett, Jr., J.H., East, E.W. and Soibelman, L. (2011). "COBIE-based Lightweight Representation of a Building Navigation Network," *The 28th International Symposium of Automation and Robotics in Construction, Seoul, South Korea*, June 29-July 2.

Treldal, N. (2008) "Integrated Data and Process Control During BIM Design Focused on Integrated Design of Energy and Indoor Climate Conditions", Master's Thesis, Department of Civil Engineering, Technical University of Denmark.

Tulke, J., Tauscher, E., Theiler, M. and Riedel, T. (2010). "Open IFC Tools.", Retrieved from http://www.openifctools.org/ Open_IFC_Tools/Home.html

Yuan L. and Zizhang H. (2008). "Indoor Navigation: a Framework of Combining BIM with 3D GIS", *44th ISOCARP Congress*.