Occupancy Analysis Using BIM and Cell-DEVS Simulation with RISE Remote Simulation (CD++ and Lopez models)

Sixuan Wang
Gabriel Wainer

Dept. of Systems and Computer Engineering
Carleton University
1125 Colonel By Dr. Ottawa
ON K1S5B6, CANADA
{swang, gwainer}@sce.carleton.ca

Keywords: BIM, IFC, Cell-DEVS, Occupancy Analysis, Copenhagen's New Elephant House

Abstract

Building Information Modeling (BIM) and its open standard Industry Foundation Classes (IFC) are becoming popular in the design phase in the Architecture and Construction industry. Here, we focus on integrating BIM and RISE (RESTful Interoperability Simulation Environment) remote simulation with DEVS (Discrete Event systems Specification) simulation of occupancy analysis. We present a case study for Copenhagen's New Elephant House, where people can move with different direction probabilities and wait randomly when visiting in the two floors of this building. The idea is to automate to extraction of building information that can be subsequently used in a remote simulation. Our occupancy model is implemented in two simulators (original CD++ version and Lopez version for multi variables/ports). Then we use RISE remotely running different simulation scenarios. We also show how to obtain advanced 3D visualization within BIM authoring tools. This work brings designers to understand better among different building properties occupancy management and future improvement.

1. INTRODUCTION

Building Information Modeling (BIM) plays an important role in the Architecture, Engineering and Construction (AEC) industry to create, document, manage and exchange information. BIM uses 3D object-oriented building modeling to realize accurate AEC projects with minimized costs, improving the way architects-contractors and fabricators work [1]. To improve the interoperability and standardization between different domains in AEC projects, buildingSMART International (former International Alliance for Interoperability, IAI) [2] proposed the Industry Foundation Classes (IFC), an international open standard for the data representation and exchange in the building industry.

Modeling and simulation (M&S) has been used in BIM to analyze the performance of building designs using an iterative process with several cycles through alternative simulations [3]. In order to evaluate the performance or find the optimal solution, the designers test and refine their solutions with different simulation scenarios. A variety of methods can be used to develop M&S applications. Here, we are interested in exploring the use of the Discrete Event System Specification (DEVS) [4] and Cell-DEVS [5] formalisms with this goal.

In this paper, we propose a uniform process for integrating BIM standard files (IFC) and DEVS/Cell-DEVS simulation. To do so, we would show how to extract information from the IFC file, run simulation and view 3D visualization. The current implementation uses CD++ [5] for Cell-DEVS toolkit, Lopez simulator for multi ports/variables, RISE [6] for a remote simulation middleware, Autodesk Revit Architecture [7] and Autodesk 3ds Max [8] for BIM tools, and BimServer.org [9] for querying the IFC file.

Recently, occupancy analysis has become a hot topic in AEC industry [10] due to the concerns of sustainability and green building. People want to achieve high-energy efficiency through performance evaluation and feedback control by simulations. Our case study is based on Copenhagen's New Elephant House [11] to simulate people moving behaviors for occupancy analysis. Visitors walk in from the main entrance on floor1, go downstairs to floor2 and then leave the house through exit. Visitors may randomly move or wait at a place for a while. This kind of application can enable designers to extract information automatically from a standard IFC file into a specific simulation model, and run the simulation. Finally, the designers can visualize the 3D simulation results to understand better the occupancy level of the New Elephant House during different scenarios, (e.g., doors location, stairs number, rush/slash hours, different movement probabilities of directions, etc.).

This paper is organized as follows: Section 2 discusses the related work in the integrating efforts for the BIM and simulation. Section 3 introduces the integrated M&S process. Section 4 discusses using IFC to extract information. Section 5 introduces the occupancy model with Lopez simulator. Section 6 represents the model with

original CD++ version. Section 7 tells the 3D visualization. Section 8 illustrates simulation results of studied Elephant House. Finally, we conclude this paper in Section 9.

2. RELATED WORK

Nowadays, BIM has been widely applied in the AEC industry for 3D-rendering, drawing extraction, estimation of cost, and emergency detection. At present, there are numerous BIM software applications, including Autodesk Revit Architecture, Autodesk 3Ds Max, Bentley Architecture, Graphisoft ArchiCAD, VectorWorks Architect and others. These BIM authoring tools facilities the workload of building design and improve performance analysis [12]. Aiming at the interoperability and standardization, Industry Foundation Classes (IFC) [4] is an open standard for exchanging BIM data in the building industry. The IFC is proposed by buildingSMART International [2]. IFC is an open standard of BIM which enables different users (such as architects, engineers, contractors, suppliers, fabricators, etc.) to build models together.

Modeling and simulation can be used in construction projects for analyzing the building design and evaluating performance. Some of the popular tools in this domain are Simphony [13] and Stroboscope [14], which simulate logical relationships between different resource locations. Cell-based analysis has been used in the construction site analysis and emergency simulation, such as [15, 16]. There is also many recent work in occupancy simulation. In [10] the authors proposed a stochastic agent-based model of occupancy dynamics in a building; the authors in [17] analyze occupancy information for energy efficiency by studying zone-level control.

Discrete Event Systems Specification (DEVS) formalism [4] has gained popularity to model a variety of problems. DEVS is a framework for constructing the discrete-event hierarchical modular systems, composed by behavioral (atomic) and structural (coupled) components.
The Cell-DEVS formalism [5] extended the DEVS formalism allowing the simulation of discrete-event cellular models. CD++ [5] is a DEVS/Cell-DEVS modeling and simulation toolkit. It is a standalone version running on a local PC, which simulation results can be seen in 2D scenes using CD++. RISE (RESTful Interoperability Simulation Environment) [6] is a simulation middleware to support RESTful-CD++ web services for the remote simulation, which aims to support interoperability and mash-ups of distributed simulations regardless of the model formalisms, model languages or simulation engines.

In recent years, we have developed different construction and architecture projects with DEVS and Cell-DEVS. In [18] we proposed a preliminary way to combine BIM and DEVS simulation, with the case of emergency evacuation, which can help analyzing bottlenecks of building design for determining an optical evacuation plan. In [19] the authors introduced Cell-DEVS models for construction sites, trying to deal with the construction performance and the crane behavior for conflict analysis. In [20], an Interactive Environment System (IES) was introduced for a basic integration between BIM and Cell-DEVS, in which a simulation of Diffusion Limited Aggregation (DLA) was used to model the growth of mold in building walls. However, most of these efforts required ad-hoc tailoring and test scenarios.

3. BIM & CELL-DEVS SIMULATION PROCESS

Performance analysis has several iterative cycles through alternative simulations. During each cycle, various management teams collaborate and share multidisciplinary design information using a number of BIM authoring tools. Figure 1 illustrates the uniform BIM and simulation process, adapted from our previous architecture mentioned in [18]. The overall architecture includes three subsystems: BIM Data Collection, DEVS Simulation and BIM 3D Visualization. This process emphasizes the significance of using the IFC standard to facilitate the collaborative development and interoperability, since most BIM authoring tools currently support the use of IFC standard files as a core information repository for sharing, collaborating and communicating.

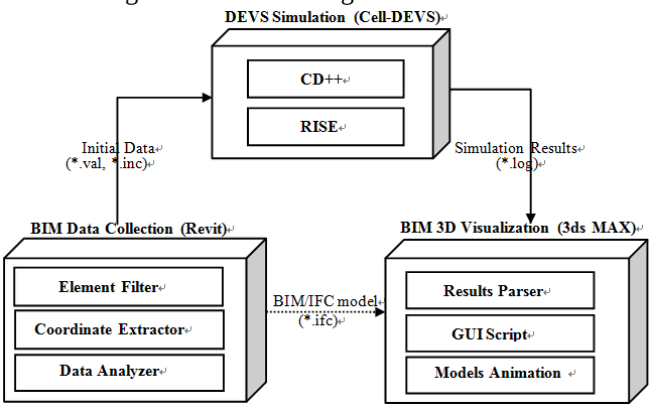

Figure 1. Uniform BIM & DEVS Simulation Process (adapted from [18])

1) BIM Data Collection: it is done automatically using the IFC standard, generating initial data files for the DEVS simulator. This includes an *Element Filter* for selective properties, and a *Coordinate Extractor* for getting (x, y, z) coordinates of filtered BIM element. The generated files are used for simulation inputs. This step will be explained in Section 3.1.

2) DEVS Simulation: it builds a simulation model based on the collected data, and then executes simulation either in standalone version *CD++* or via *RISE* middleware remotely. Model details are discussed in Section 3.2. In RISE, there are various CD++ versions available, including DCDpp for normal and distributed simulation and Lopez with different port/variable values.

3) BIM 3D Visualization: it visualizes the simulation results in 3D, based on IFC standard file, providing an

intuitive mean for analysis. It parses the simulation log files, and loads it and BIM file using *GUI Script*. The results are generated by the *Models Animation*. Section 3.4 would explain this step.

As a case study, we will employ this method analyzing occupancy levels of the Copenhagen's New Elephant House, using Autodesk Revit Architecture [7] for the BIM Data Collection, and Autodesk 3Ds Max [8] for BIM 3D Visualization. The Copenhagen's New Elephant House (corresponding IFC building can be seen in Figure 2) opened in June 2008 replacing a structure dating from 1914. It is located in The Copenhagen Zoo, the largest cultural institution in Denmark, attracting over 1.2 million visitors a year. The New Elephant House seeks to create a close visual relationship between the zoo and the park, which has two floors. Floor 1 exhibits related scientific knowledge or artwork; while on floor 2, visitors can see elephants close on both sides. Visitors walk in from the main entrance on floor1, go downstairs to floor2 and then leave the house through exit. Visitors randomly move following the pathway or wait for a while by random time within hot zones.

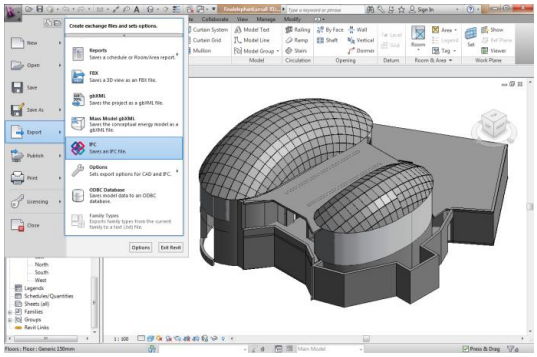

Figure 2. Copenhagen's New Elephant House in BIM

For the random movement, each floor has guideline routes or visiting pathway for the visitors moving forward orderly. However, it is not mandatory that people follow this guide. Normally, they can move to the surrounding eight directions with different probabilities. Figure 3 illustrates the relationship of potential probabilities in the eight directions, the darker the neighbor is, the more likely a visitor would move. In our case, assume there is visitors standing on a pathway to move right, they would move by a chance of 70% forward (%F), 8% left-forward (%LF), 8% right-forward (%RF), 4% left (%L), 4% right (%R), 2% left-back (%LR), 2% right-back(%RB), and 2% back (%B). These probabilities can be evaluated from statistic and set as macro variables before the simulation running.

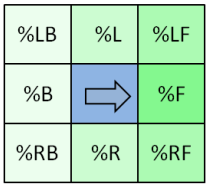

Figure 3. People moving with different probabilities

For random wait, the moving speed of each visitor is also not a constant. It takes different time for each visitor to move to the next place. Especially at some spots where they are interesting, they would stop or even sit down there.

4. Data Collection

In order to get the needed initial data for simulation, we need to extract information from BIM files. Most BIM tools support the IFC standard files for the interoperability. The most abstract entity in the IFC model is *IfcRoot*, which is in the kernel and provides attributes of identification, ownership and self-description for all sub-entities. Figure 4, summarized from [21], is a hierarchical diagram of a part of the EXPRESS standard under *IfcRoot*. For the case of the Elephant's House, the initial information for simulation purpose is stored in *IfcWall* and *IfcDoor*. Both of them are subclasses of *IfcElement* and contain coordinates information either in *IfcLocalPlacement* or in *IfcProductDefinationShape*. *IfcLocalPlacement* has the attribute of *IfcAxis2Placement3D* with *IfcCartesionPoint* (coordinates) and *IfcDirection* (the trend of spreading). *IfcProductDefinationShape* could contain *IfcPolyline* with the explicit geometric boundary points of its surfaces.

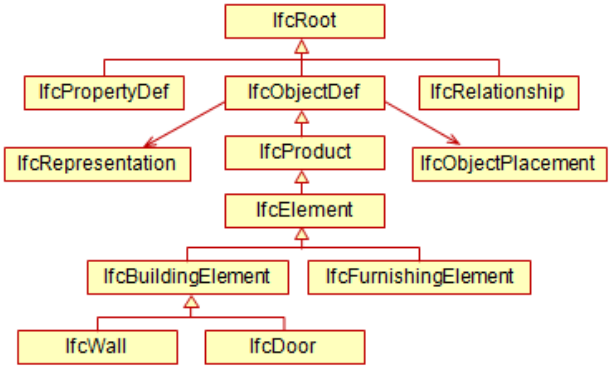

Figure 4. *IfcWall* and *IfcDoor* in IFC hierarchy

As mentioned in [18], we need two main steps: 1) collecting the necessary data from IFC by Element Filter and Coordinate Extractor; 2) generating initial data files for Cell-DEVS by Data Analyze.

For the first step, we use open APIs from BIMServer.org [9] for querying and filtering the needed IFC elements. BIMserver.org is a model-driven open-source tool that adopts EMF to represent the IFC data. We query all *IfcWall* and *IfcDoor* in the two Floors, obtaining the corresponding (x, y, z) coordinates of the filtered BIM elements. Note that *IfcDoor* can be classified into entrance, stairs and exits; and we could query other useful element type for future extensions (e.g, furniture, windows, etc.). For the second step, based on the filtered information, we can build the layout of each floor and generate specific initialization data files for the CD++ simulation. For instance, in this occupancy models, we first calculate the scale of each floor, converting it to a unified size: XMIN is the horizontal ordinate of the left-most point, while XMAX

represents the horizontal ordinate of the right-most point. Similarly, we get YMIN and YMAX for the vertical ordinates. Then, knowing the size of each cell, we can compute the number of cells in each of the three dimensions, and generate the simulation rules for the CD++simulator. Finally, we calculate the Elephant House size (10x22 cells per floor) and the layout information (wall, entrance, exit, stairs).

5. Occupancy Model (Lopez on RISE with multi variables/ports)

We use Cell-DEVS to simulate the behavior of a three-dimensional object representing the studied building. For the simulator of Cell-DEVS, we choose Lopez simulator on RISE Web Services. As seen in Figure 5, from the top view, each floor has 10x22 cells, and each cell represents a square place associated with physical horizontal coordinates. There are 2 floors, connecting with stairs. For example, Cell (3, 0, 0) represents the entrance, Cell (8, 21, 1) represents the stair going downstairs, etc. Each cell has five state variables: Movement, Phase, Pathway, Layout, and Hotzone.

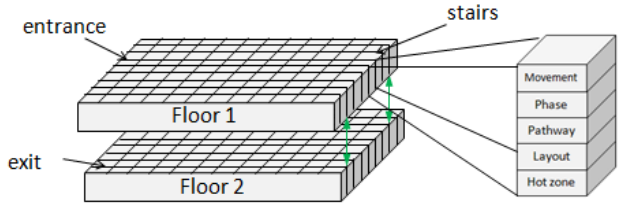

Figure 5. Four-dimensional cells

Table 1 lists all cell states, categorized by the five state variables.

Table 1. Cell States

Cell State	State Name	Cell State	State Name
Variable 1 : Movement		Variable 2 : Phase	
0	Not occupied	1	Intent
1	Occupied	2	Grant
10-18	Intent direction	3	Wait
20-28	Intended cell get grant	4	Move
40-48	Target cell grant one	Variable 3 : Pathway	
30-38	Intended cell can move	5	Right
39	Intended cell cannot move	6	Up
Variable 4 : Layout		7	Left
0	Space	8	Down
2	Wall	Variable 5 : Hotzone	
3	Entrance	0	No wait
4	Exit	1	Wait 0-1 cycle
3.1	Stair(Floor1)	2	Wait 0-2 cycles
3.2	Stair(Floor2)	3	Wait 0-3 cycles

- **Movement**: 0 means a cell that is not occupied by anyone, 1 means the cell is currently occupied, it also records other states related to the four phases, reflecting the relationship with the neighbors.
- **Phase**: each movement cycle has four phases (Intent, Grant, Wait, Move), details are discussed in Section 5.1
- **Pathway**: shows the visiting routes. Visitors tend to move following the pathway with certain probabilities. 5 means Right direction, 6 means Up direction, 7 means Left direction and 8 means Down direction.
- **Layout**: consists of space (0), wall (2), entrance (3), stairs (3.1 for upstairs and 3.2 for down stairs), exit (4), etc.
- **Hot Zone**: reflects popularity levels of the spots influencing different potential waiting time. The higher value the hot zone is, the higher the probability that a visitor in the hot zone would stay.

In this model, people moving is based on an expended Moore and Von Neumann neighborhood (see Figure 6), with allowance for larger neighborhoods. The basic neighborhoods are nine Moore neighbors ((-1,-1,0)…(1,1,0)), each neighbor indicates one direction from E, NE, N, NW, W, SW, S, SE or central. To determine the movement from stairs, we add two other neighbors (0,0,-1) for going downstairs and (0,0,1) for receiving people from upstairs.

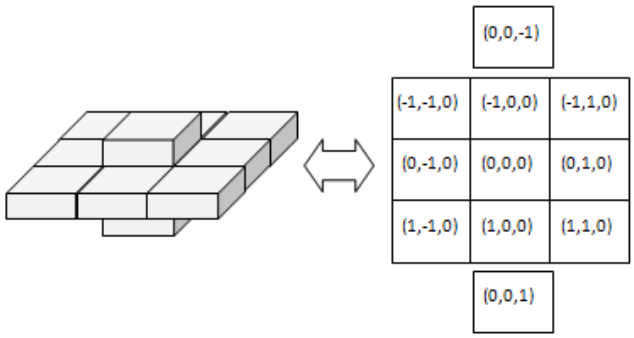

Figure 6. Neighborhood

Layout layer contains the building information from the studied building, which is extracted from BIM Data Collection process through the standard IFC files. The reason to separate this layout layer from other layers is for better abstraction and modularity, which enable the designer operate and extend this lever easier in the future design. In the Elephant House, it has two floors, and we are only interested in outside walls, inside pillar walls, entrances, exits and stairs toward up/down.

The pathway is the guideline routes for visitors moving. The exhibition rooms may have specific pathway to control the people moving flow. Normally, the pathway points to the shortest path towards the exit. In our case, we overlay a Voronoi diagram of the route to an exit or stair. Figure 7 demonstrates how the pathway would like. This pathway can be initialized iteratively by waving from the cells nearby exit/stairs to the distant cells. The implemented rules would run at first when simulation running.

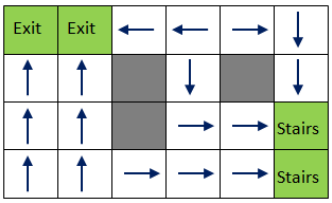

Figure 7. Pathway initialization

Hot zones affect potential staying/waiting time. As shown in Figure 8, assume there are elephant herds closely outsides the bottom area, people near the middle of bottom would more likely to stay watching than the people far away do. Therefore, there are different hot zone values that indicate the distance to some hot spots. The higher value the hot zone is, the more likely visitors would stay. For example, the people in a dark red cell would randomly wait there from 0 to 4s; while the people in a light red cell would just stay there no more than 1s.

1	1	1	1
2	2	2	2
3	3	3	3
3	4	4	3

Figure 8. Hot zones

5.1. Movement Phases Implementation (Lopez with multi variables/ports)

In the lopez simulator for Cell-DEVS, a state variable can be implemented by either variable or port. The difference is that variables can only be used by its owned cell, never notify to neighbors; while port values can be known to neighbors. In our implementation, ports are Movement, Phase and Pathway; while variables are Layout and Hotzone.

In order to realize random movement and random waiting, the movement behavior is divided into four phases (*Intent*, *Grant*, *Wait*, and *Move*), which state diagram is shown in Figure 9.

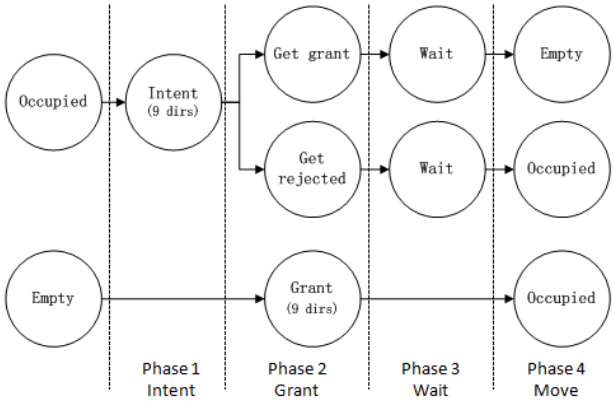

Figure 9. State diagram of Movement

Generally, for an occupied cell, a visitor chooses a direction randomly at the *intent* phase. If the target cell accepts it, it changes to *get grant*; otherwise, it turns to *get rejected*. If granted, the visitor would *wait* for some time randomly according to the hot zone where the visitor is standing at, and then *empty* the cell at the *move* phase. If rejected, the visitor only needs to wait one phase time to turn to the move phase and keeps occupied. For an *empty* cell, the logic is much simpler. It would choose a surrounding intended cell at the *grant* phase and would change to *occupied* at the *move* phase.

At the beginning of the simulation, visitors would be initialized at the main entrances with certain probability (*VisitorRate*), in order to mimic different input flow rates with rush/slash hour during the opening time. In the current implementation, each cycle has 4s (each phase has 1s). We firstly check whether it is the beginning of the cycle *(remainder(time, 4)=0)*, then generate a person at each entrance of Floor 1.

```
rule : {~movement := 2; ~phase := 1;} 0
{uniform(0,1)<#Macro(VisitorRate)    and
remainder(time, 4)=1 and (0,0,0)~phase =
0 and (0,0,0)~movement=0 and $layout=3}
```

Phase 1: Intent

During this phase, the intent direction is determined by two factors: the pathway direction and the direction probability. We first check if it is in the intent phase, and if the cell is a stair and the cell below is empty, it gets the intent direction of 10. If the cell is not a stair, according to the pathway value, we find the probability distribution of directions (see Figure 10, which splits 100 into 8 pies). For example, if the pathway is 6 (up), we would use the first chart of Figure 10. Then we generate a random number between 0 and 100, and check which direction that the random number is located. For example, if the random number is 58, it is in *%F (0-69)*, so we get the intent direction to go up.

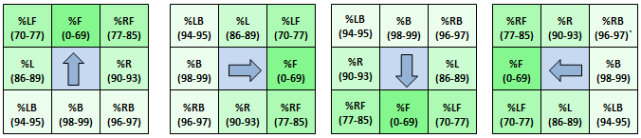

Figure 10. Intent probability distribution with the pathway

At last, the cell changes to 10-18, which unit value corresponds with the intent direction: D(0), E(1), NE(2), N(3), NW(4), W(5), SW(6), S(7), SE(8) (see left part of Figure 11). E.g., for going up, it should be 13. Note here we do not care whether the target cell is available, it will be checked in the following phases.

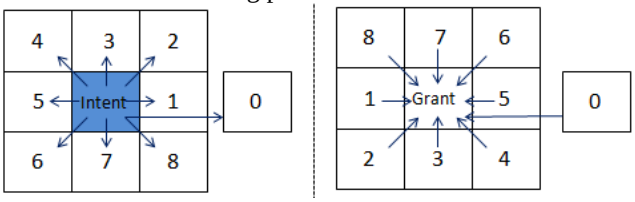

Figure 11. Intent & reverse Grant direction values

To implement the *Intent* phase, we use following rules:

```
rule : {~movement := 10; ~phase := 2;} 1
{ (0,0,0)~phase = 1 and (0,0,0)~movement
= 1 and $layout = 5}
rule : {~movement := uniform(0,1);
~phase := 1.1;} 0 (0,0,0)~phase = 1 and
(0,0,0)~movement=1 and
(0,0,0)~pathway>=5} ...
rule : {~movement := 11; ~phase := 2;} 0
(0,0,0)~phase = 1.1 and (0,0,0)~pathway
= 5 and (0,0,0)~movement > 0.0 and
(0,0,0)~movement <= #Macro(Front) } ...
rule : {~movement := 18; ~phase := 2;} 0
(0,0,0)~phase = 1.1 and (0,0,0)~pathway
=8 and (0,0,0)~movement > #Macro(Front)
+#Macro(Left-Front) and (0,0,0)~movement
<=#Macro(Front)+...+#Macro(Right-Front)}
```

Phase 2: Grant

After the occupied cell chose its intended direction, more than one person may want to enter into a same cell (collision). To handle this problem, each empty cell will choose only one neighbor, and change its state. In detail, we should make sure it is the grant phase, and if the cell is stair and the cell above wants to come down (10), so it changes to 40 and phase 4 to move. Otherwise, it chooses an intent surrounding cell in order, and changes to 41-48, which unit value corresponds with one of the reverse eight directions shown in the right part of Figure 11). E.g., 41 means the current cell accepts the left neighbor to come in. To keep coherence, the cells with intent direction (10-18) change to 20-28 and phase 3 for waiting.

The rules for the *Grant* phase are like as follows:

```
rule : {~movement := 40; ~phase := 4;} 1
{ (0,0,0)~movement = 0 and (0,0,-
1)~movement = 10 }
rule : {~movement := 41; ~phase := 4;} 1
{ (0,0,0)~movement = 0 and (0,-
1,0)~movement = 11 and $layout != 2} ...
rule : {~movement := 48; ~phase := 4;} 1
{ (0,0,0)~movement = 0 and (-1,-
1,0)~movement = 18 and $layout != 2}
rule:{~movement := ((0,0,0)~movement+10);
~phase := 3;} 1 { (0,0,0)~movement >= 10
and (0,0,0)~movement <= 18 }
```

Phase 3: Wait

We use this phase to mimic the behavior of random wait. If a person is granted by the target cell (he has an intent direction and the target cell chose him), he will stay there for a random amount of time according to the hotzone where he is standing. We implement this by adding different delays in the associated rules. We first get a random number from 0 to the corresponding hotzone value (e.g., 2 means waiting for 2 cycles); because each cycle has 4s, plus 1s for the forth *Move* phase, it actually would wait 4*N+1 (9s) from 20-28 to change to 30-38, which unit value corresponds with the intent direction. Note that the stair (20) has the same rule about waiting as the normal cells. Otherwise, if not granted, the person cannot move anywhere and he should try again for a next moving cycle; so it changes to 39 and will only delay 1s for the forth *Move* phase. Take the above as an example, if the cell is 21 (want to go right) and its right cell is 41, it will wait there for 9s to change to 31.

The rules of this phase are like this:

```
rule : {~movement := 30; ~phase := 4;} 1
{ (0,0,0)~phase = 3 and (0,0,0)~movement
= 20 and (0,0,1)~movement = 40 }
rule : {~movement := 31; ~phase := 4;}
{ 1 + 4*randInt($hotzone) }
{ (0,0,0)~phase = 3 and (0,0,0)~movement
= 21 and (0,1,0)~movement = 41 }
rule : {~movement := 38; ~phase := 4;}
{ 1 + 4*randInt($hotzone) }
{ (0,0,0)~phase = 3 and (0,0,0)~movement
= 28 and (1,1,0)~movement = 48 }
rule : {~movement := 39; ~phase := 4;} 1
{ (0,0,0)~phase = 3 and
(0,0,0)~movement >=20 and
(0,0,0)~movement <= 28 }
```

Phase 4: Move

Now, every intended cell is 30-38 (*granted*) or 39 (*rejected*), so the granted visitor can move to the target cell. To finish the moving for next cycle, we empty the intended cells that are granted to 0, and the rejected ones to 1. For the target cell, it changes from 40-48 to 1 after checking its corresponding intended cell is still existing.

Here are the rules for this part:

```
rule : {~movement := 0; ~phase := 1;}
100 { (0,0,0)~phase = 4 and
(0,0,0)~movement = 30 and
(0,0,1)~movement = 40 } ...
rule : {~movement := 1; ~phase := 1;}
100 { (0,0,0)~phase = 4 and
(0,0,0)~movement = 48 and (-1,-
1,0)~movement = 38}
```

So far, we have discussed entrances, stairs and normal cells. The last thing is to consider the exit, we just need to empty it if it is occupied, the rule of exit is:

```
rule : {~movement := 0;} 100 { (0,0,0) =
1 and $layout = 4 and (0,0,0)~movement=1}
```

Here is an example to show the movement cycle (see Figure 12). Initially, there were two cells occupied. After phase1 at 1s, two persons randomly intended to move to a same cell (one with 11 and the other with 12). After the phase at 2s, the target cell chose the cell left (it changed to 41). In phase 3, the granted cell waited for 1 more cycle (1+4=5s). After this wait expired, the time reached at 7s. At last, the granted cell moved to its target, while the other cell kept staying there.

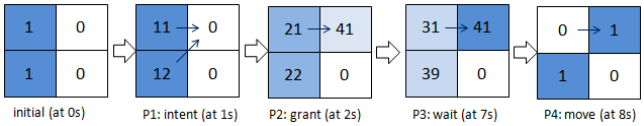

Figure 12. Movement Phases example

6. Occupancy Model (original CD++ version)

This model originally is implemented in CD++ version, which is run as standalone. The implementation is very similar as Lopez version mentioned in Section 5. In order to represent different state variables of each cell, we add a fourth dimension to each cell to represent: Movement, Pathway, Layout, and Hot zone.

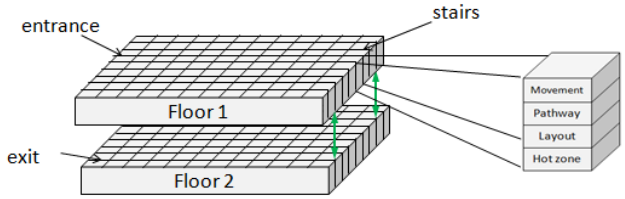

Figure 13. Four-dimensional cells

Table 2 lists all cell states. The four variables are stored in different layers in the fourth dimension of our cell space.

Table 1. Cell States

Cell State	State Name	Cell State	State Name
Layer 4 : Movement		Layer 2 : Layout	
0	Not occupied	0	Space
1	Occupied	2	Wall
10-18	Intent direction	3	Entrance
20-28	Grant: intent cell	4	Exit
40-48	Grant: target cell	3.1	Stair(Floor1)
30-38	Wait: can move	3.2	Stair(Floor2)
39	Wait: cannot move		
Layer 3 : Pathway		Layer 1 : Hot zone	
5	Right	0	No wait
6	Up	1	Wait 0-1 cycle
7	Left	2	Wait 0-2 cycles
8	Down	3	Wait 0-3 cycles

- **Movement**: 0 means a cell that is not occupied by anyone, 1 means the cell is currently occupied, it also records four phases during each movement cycle (Intent, Grant, Wait, Move), reflecting the relationship with the neighbors.
- **Pathway**: shows the visiting routes. Visitors tend to move following the pathway with certain probabilities. 5 means Right direction, 6 means Up direction, 7 means Left direction and 8 means Down direction.
- **Layout**: consists of space (0), wall (2), entrance (3), stairs (3.1 for upstairs and 3.2 for down stairs), exit (4), etc.
- **Hot Zone**: reflects popularity levels of the spots influencing different potential waiting time. The higher value the hot zone is, the higher the probability that a visitor in the hot zone would stay.

In this model, people moving is based on an expended Moore and Von Neumann neighborhood (see Figure 14), with allowance for larger neighborhoods. The basic neighborhoods are nine Moore neighbors ((-1,-1,0,0)…(1,1,0,0)), each neighbor indicates one direction from E, NE, N, NW, W, SW, S, SE or central. To determine the movement from stairs, we add two other neighbors (0,0,-1,0) for going downstairs and (0,0,1,0) for receiving people from upstairs. In order to know the variable information of current considering cell, we should know pathway (0,0,0,1), layout (0,0,0,2), and hotzone (0,0,0,3). For using rules to determine the initial pathway automatically, we add other four Von Neumann neighbors for pathway layer ((0,1,0,1)…(-1,0,0,1)) to present the direction from E, N, W, S.

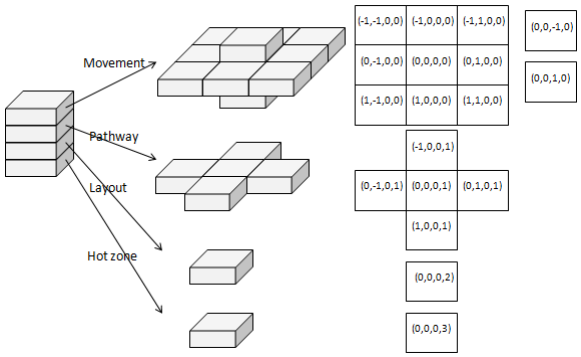

Figure 14. Neighborhood

6.1. Movement Phases Implementation (original CD++ version)

Same as mentioned in Section 5.1, the implementation is similar to lopez version. The movement behavior is divided into four phases (*Intent*, *Grant*, *Wait*, and *Move*).

At the beginning of the simulation, visitors would be initialized at the main entrances with certain probability, in order to mimic different input flow rates with rush/slash hour during the opening time. In the current implementation, each cycle has 4s (each phase has 1s). We firstly check whether it is the beginning of the cycle *(remainder(time, 4)=0)*, then generate a person at each entrance of Floor 1.

```
rule : {1} 4 { remainder(time, 4)=0 and
(0,0,0,0)=0    and    (0,0,0,2)=3    and
(0,0,0,1)>=5 }
```

Phase 1: Intent

During this phase, the intent direction is determined by two factors: the pathway direction and the direction probability. We first check if it is in the intent phase, and if the cell is a stair and the cell below is empty, it gets the intent direction of 10. If the cell is not a stair, according to the pathway value, we find the probability distribution of directions (see Figure 10, which splits 100 into 8 pies). For example, if the pathway is 6 (up), we would use the first chart of Figure 10. Then we generate a random number between 0 and 100, and check which direction that the random number is located. For example, if the random number is 58, it is in *%F (0-69)*, so we get the intent direction to go up. At last, the cell changes to 10-18, which unit value corresponds with the intent direction: D(0), E(1), NE(2), N(3), NW(4), W(5), SW(6), S(7), SE(8) (see left part of Figure 11). E.g., for going up, it should be 13. Note

here we do not care whether the target cell is available, it will be checked in the following phases.

To implement the *Intent* phase, we use following rules:

```
rule : {10} 1 { remainder(time, 4)=0 and
(0,0,0,0)=1 and (0,0,0,2)=3.1}
rule : {uniform(0,1)} 1 { remainder(time,
4)=0 and (0,0,0,0)=1 and (0,0,0,1)>=5}
rule : {11} 0 { (0,0,0,1)=5 and
(0,0,0,0)>0.0 and (0,0,0,0) <=
#Macro(Front)...
rule : {14} 0 { (0,0,0,1)=6 and
(0,0,0,0) > #Macro(Front) and (0,0,0,0)
<= #Macro(Front) + #Macro(Left-Front) ...
rule : {16} 0 { (0,0,0,1)=7 and
(0,0,0,0) > #Macro(Front) + ... +
#Macro(Right-Front) ...
rule : {18} 0 { (0,0,0,1)=8 and
(0,0,0,0) > #Macro(Front) + ...
+#Macro(Right-Front) }
```

Phase 2: Grant

After the occupied cell chose its intended direction, more than one person may want to enter into a same cell (collision). To handle this problem, each empty cell will choose only one neighbor, and change its state. In detail, we should make sure it is the grant phase, and if the cell is stair and the cell above wants to come down (10), so it changes to 40. Otherwise, it chooses an intent surrounding cell in order, and changes to 41-48, which unit value corresponds with one of the reverse eight directions shown in the right part of Figure 11). E.g., 41 means the current cell accepts the left neighbor to come in. To keep coherence, the cells with intent direction (10-18) change to 20-28.

The rules for the *Grant* phase are like as follows:

```
rule : {40} 1 { remainder(time, 4)=1 and
(0,0,0,0)=0 and (0,0,-1,0)=10 }
rule : {41} 1 { remainder(time, 4)=1 and
(0,0,0,0)=0 and (0,-1,0,0)=11 and
(0,0,0,2)!=2}...
rule : {48} 1 { remainder(time, 4)=1 and
(0,0,0,0)=0 and (-1,-1,0,0)=18 and
(0,0,0,2)!=2}
rule : {(0,0,0,0)+10} 1 { remainder(time,
4)=1 and (0,0,0,0)>=10 and
(0,0,0,0)<=18 }
```

Phase 3: Wait

We use this phase to mimic the behavior of random wait. If a person is granted by the target cell (he has an intent direction and the target cell chose him), he will stay there for a random amount of time according to the hotzone where he is standing. We implement this by adding different delays in the associated rules. We first get a random number from 0 to the corresponding hotzone value (e.g., 2 means waiting for 2 cycles); because each cycle has 4s, plus 1s for the forth *Move* phase, it actually would wait 4*N+1 (9s) from 20-28 to change to 30-38, which unit value corresponds with the intent direction. Note that the stair (20) has the same rule about waiting as the normal cells. Otherwise, if not granted, the person cannot move anywhere and he should try again for a next moving cycle; so it changes to 39 and will only delay 1s for the forth *Move* phase. Take the above as an example, if the cell is 21 (want to go right) and its right cell is 41, it will wait there for 9s to change to 31.

The rules of this phase are like this:

```
rule : {30} 1 { remainder(time, 4)=2 and
(0,0,0,0)=20 and (0,0,1,0)=40 }
rule : {31} { 1 + 4*randInt((0,0,0,3)
+1) } {remainder(time, 4)=2 and (0,0,0,0)
= 21 and (0,1,0,0) = 41 } ...
rule : {38} { 1 + 4*randInt((0,0,0,3)
+1) } {remainder(time, 4)=2 and (0,0,0,0)
= 28 and (1,1,0,0)=48 }
rule : {39} 1 { remainder(time, 4)=2 and
(0,0,0,0)>=20 and (0,0,0,0)<=28 }
```

Phase 4: Move

Now, every intended cell is 30-38 (*granted*) or 39 (*rejected*), so the granted visitor can move to the target cell. To finish the moving for next cycle, we empty the intended cells that are granted to 0, and the rejected ones to 1. For the target cell, it changes from 40-48 to 1 after checking its corresponding intended cell is still existing.

Here are the rules for this part:

```
rule : {0} 1 { remainder(time, 4)=3 and
(0,0,0,0)>=30 and (0,0,0,0)<=38 }
rule : {1} 1 { remainder(time, 4)=3 and
(0,0,0,0)=39 }
rule : {1} 1 { remainder(time, 4)=3 and
(0,0,0,0)>=40 and (0,0,0,0)<=48}
```

So far, we have discussed entrances, stairs and normal cells. The last thing is to consider the exit, we just need to empty it if it is occupied, the rule of exit is:

```
rule : {0} 1 { (0,0,0,2)=4 and
(0,0,0,0)=1}
```

7. 3D Visualization

3D visualization provides a more intuitive and attractive way to obtain visual simulation results in BIM authoring tools, enabling the designers to check the building performance and people behaviors under different properties. Most BIM authoring tools support full-featured 3D visualization of building. Among them, Autodesk 3ds Max is a powerful BIM tool for its animation and rendering ability for 3D visualization. The IFC file can easily imported into 3ds Max, what we want is to view the simulation results into the model that can be visualized in 3ds Max. To do this, we have developed an advanced visualization tool in 3ds MAX, which is upgraded based on our existing version. This new tool expanded new functionalities for reusability and scalability, including a parser program using Python script (to parse simulation results log file), expended GUI in 3ds Max with extendable script. The GUI (see Figure 13) provides several options of hiding different building floors for visibility and filtering models in a small area for the optimization purpose. We

then added new animation features of models: 1) arrow models with key framing ability and 2) realistic models to animate real body movement using Motion Mixer. This work brings better visualization of simulation results, enabling the designers to check the simulation results of BIM, find the flaws and plan for improvement.

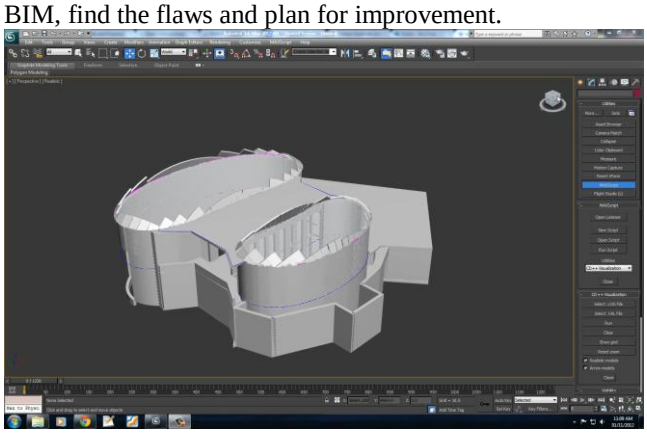

Figure 13. 3ds Max visualization user interface.

8. SIMULATION RESULTS

In this section, we discuss some simulation results, in order to analyze occupancy levels or find out bottlenecks. They are generated from the occupancy model defined in Section 3, using different building designs properties (doors location, stairs number, incoming rate, hot zones, moving probabilities, etc.). All the tests are running under 10 mins duration of the visiting hours of the elephant house, which has 150 cycles of movements (each cycle costs 4s).

Our initial test (see Figure 14) shows the basic behavior of visitors under normal properties during the rush hours. The house has two floors, people (blue) coming and getting out the house by the doors (dark marks), and going downstairs (write marks) from Floor 1(left part of each graph) to Floor2 (left part of each graph). Each cycle, according to the different phases, each person tends to wait or move randomly. During rush hours, the house opens four entrances, and each entrance generates one visitor during each cycle. In Figure 14, the blue cells represent visitors (light blue cells are waiting visitors, and each green cell represents which cell a waiting visitor wants to go). At 10 min, there are 35 visitors over 90 spaces in Floor1 and 24 visitors over 152 spaces in Floor2. Therefore, the occupancy levels are 38.9% for Floor1 and 15.8% for Floor2 respectively.

To study the impact of door location/stairs number in terms of occupancy, we discuss two modifications to the original design (see Figure 14, showing at the time of 5min and 10min results for each modification). The first one considers changing the entrances location: we separate the four gates into two parts, (two move above, while the other two move below). In the end of 10 min, the occupancy levels are 36.7% for Floor 1 and 17.1% for Floor2, which are only slight different compared with the ones in the basic case analyzed above (occupancy of Floor1 is still as twice as large of the one of Floor 2). This similarity indicates that even though doors location has changed, doors still generated the people at similar rates and no significant confliction happens.

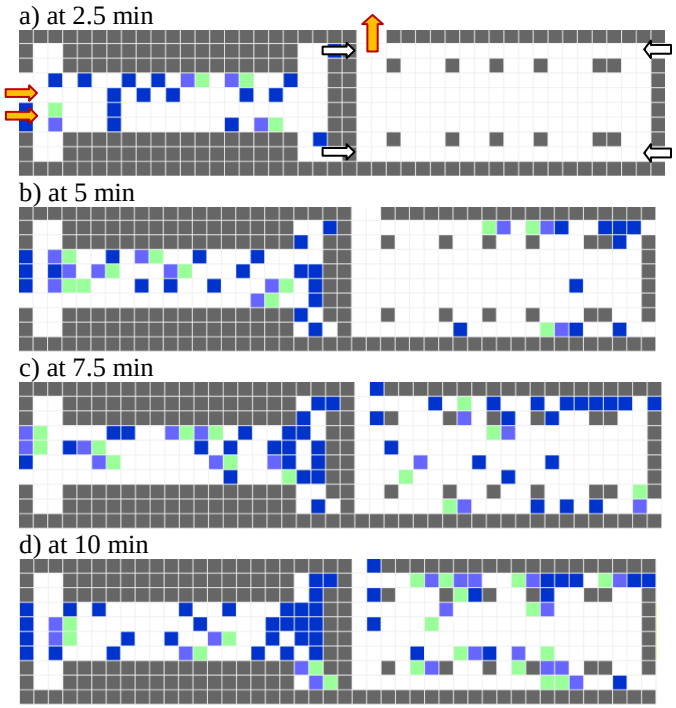

Figure 14. Simulation results of basic properties at different simulation times.

The second test in Figure 15 shows the impact of the number of stairs. We modified the model again, and added two additional staircases connecting the Floor1 and Floor2. In the end of 10 min, the occupancy levels are 25.6% for Floor 1 and 20% for Floor2, which has significant difference as before. The occupancy levels of the floors approach equal. The potential reason would be congestions happened at the stairs of Floor 1 in the above two cases, adding stairs eliminates the congestion on Floor1 and speeded up the people moving to Floor2.

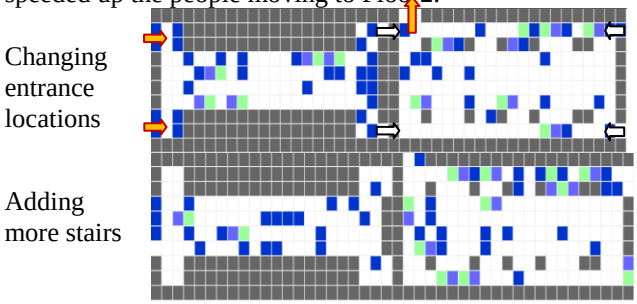

Figure 15. Simulation results of two modified properties (exit location changed / more doors added).

Some other comparative trials have been tested based on this Elephant House (see Figure 16). Totally, we change one property among hot zones, entrances location, movement direction probabilities, coming rate, stairs

number, keep other properties unchanged as the basic configuration. Main purpose is try to find which property is most significant in term of occupancy levels, so it would give the designer useful feedback to improve the maintenance and management of this house.

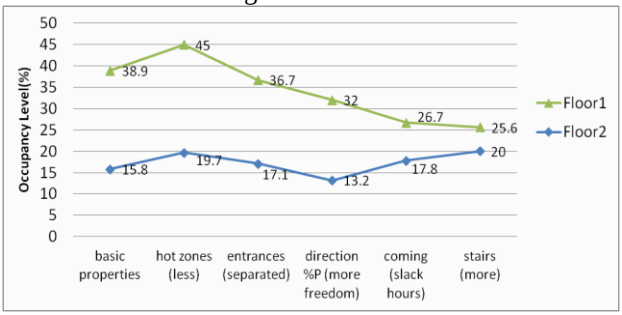

Figure 16. Comparisons of different scenarios.

In a short, coming rate and stairs number affect the occupancy level more significantly than other properties do.

1) Hot zones: we decreased the probability of people waiting by 1 cycle to mimic fast movement speed. E.g., assume in basic test a cell waits for 3 cycles, then in the new test it just waits for 2 cycles. The result shows both occupancies increase relatively obviously, which indicates the influence of people movement speed to the occupancy.

2) Movement direction probabilities: In basic properties, visitors tend move forward at rate of 70%, while in this new test, we change the forward probability to 50% to give visitors more freedom moving other directions. The results shows only a slight difference compared to original setting, which indicates the direction probabilities do not affect occupancy a lot. The possible reason would because people still can reach the stairs/exit as a normal rate.

3) Coming rate: The basic test is under hot hours. In the new test, to mimic slack hours, we prolonged the interval between two incoming visitors (2 cycles generate one visitor). The results showed significant decrease of Floor 1 (from 38.9% to 26.7%). These results indicate the coming rate of different hours during visiting time affects the occupancy level a lot. The reason of only a small change of Floor 2 would be that the rate of people coming to Floor 2 still keeps at a steady full rate, even though we reduce the incoming rate (but congestion on Floor1 has eased compared to the basic model).

We now show the occupancy simulation results using our 3D developed visualization Tool (see Figure 17). To do so, we firstly parsed the log file that generated from simulation. Then we load the BIM Elephant house building (which is IFC standard) in our tool and start the GUI. Figure 17 shows some visualization results of different options. This work brings better visualization of simulation results, enabling the designers to check the simulation results of BIM, compare the performance under different configurations and find the flaws for future improvement. Figure 17 also illustrates the ability for the designers to see different perspectives and choose among different options when using our developed tool. The tool supports two different animation models: realistic models realistic models to animate real body movement using Motion Mixer (see part A and B), and arrow models with key framing ability (see part C and D). In addition, we can hide different building floors for visibility (see part B and C) and focus on only one person (see part A).

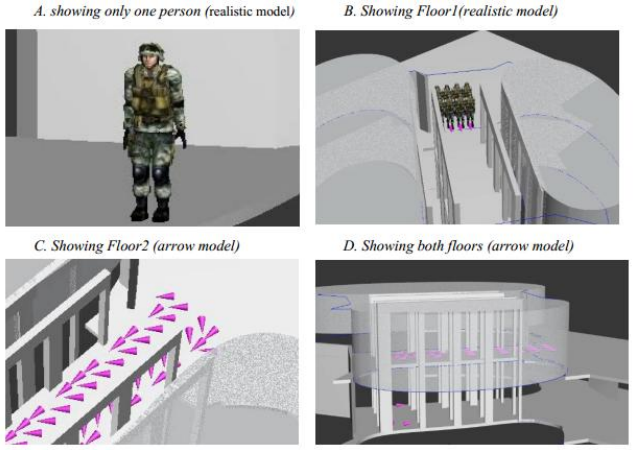

Figure 17. Different options using developed Tool

9. CONCLUSION

We propose a solution of a uniform Building Information Molding and Simulation process. We showed how to extract information from the IFC file, run simulation using Cell-DEVS and view advanced 3D visualization in 3Ds Max. Our case study uses a novel model of occupancy analysis for Copenhagen's New Elephant House. This model simulates people moving behaviors in Copenhagen's New Elephant House. Visitors walk in from the main entrance on floor1, go downstairs to floor2 and then leave the house through exit. Visitors may randomly move or wait at a place for a while. This simulator can benefit designers to understand better occupancy levels of the New Elephant House during different scenarios, (e.g., doors location, stairs number, rush/slash hours, different movement probabilities of directions, etc.). This work brings designers to understand better the significances among different building properties in order to facility the occupancy management or design suggestions for the future improvement. Some further directions of this work are to improve occupancy model by using multiple state variables version of CD++ provided in RISE WSs; as well as to study more about IFC-based BIM and the Cell-DEVS simulation.

ACKNOWLEDGEMENT

We would thank Victor Freire for his effort on upgrading the 3D visualization tool for advanced realistic models and several practical options.

REFERENCES

[1] Hardin, B. 2009. BIM and Construction Management: Proven Tools, Methods, and Workflows. Wiley.

[2] BuildingSmart. 2012. Accessed Nov 6. http://buildingsmart.com/

[3] Jiang , Y., J. Ming, D. Wu, J. Yen, P. Mitra, J. I. Messner and R. Leicht. 2012. Bim Server Requirements to Support the Energy Efficient Building Lifecycle. In Proc. 2012 ASCE international conference on computing in civil engineering.

[4] Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. Theory of Modeling and Simulation. Academic Press.

[5] Wainer , G.. 2009. Discrete-event Modeling and Simulation: a Practitioner's Approach. CRC/Taylor & Francis.

[6] Al-Zoubi, K., and G. Wainer. 2009. Using REST Web Services Architecture for Distributed Simulation. In: Proceedings of Principles of Advanced and Distributed Simulation (PADS 2009). pp. 114-121. Lake Placid, New York, USA.

[7] AutoDesk. 2012. "Autodesk Revit Architecture." Accessed Nov 6. http://usa.autodesk.com/revit-architecture/

[8] AutoDesk. 2012. "Autodesk 3ds Max." Accessed Nov 6. http://usa.autodesk.com/3ds-max/

[9] BimServer.org. 2012. Accessed Nov 6. http://bimserver.org/

[10] Liao, C., and P. Barooah. 2011. "A novel stochastic agent-based model of building occupancy." American Control Conference (ACC), 2011. IEEE.

[11] Brady, P., 2008 "The Copenhagen Elephant House: A Case Study of Digital Design Processes", in Silicon and Skin, Proceedings of the ACADIA 2008 Conference.

[12] Mihindu, S. and Y. Arayici. 2008. "Digital Construction through BIM Systems will Drive the Re-engineering of Construction Business Practices." In International Conference on Visualization, London, UK.

[13] Hajjar, D., and S.M. AbouRizk. 2002. "Unified Modeling Methodology for Construction Simulation." Journal of Construction Engineering and Management- ASCE. 128(2):174-185.

[14] Martinez, J.C. 2001. "EZStrobe - General-Purpose Simulation System Based on Activity Cycle Dia-grams." In Proceedings of 2001 WSC. Piscataway, Washington, DC. 1556-1564. IEEE.

[15] Pelechano, N., and A. Malkawi. 2008. "Evacuation Simulation Models: Challenges in Modeling High Rise Building Evacuation with Cellular Automata Approaches." Journal of Automation in Construction. Elsevier. 17(4):377-385.

[16] Yang, L.Z., D.L. Zhao, J. Li, and T.Fang. 2005. "Simulation of Kin Behavior in Building Occupant evacuation based on Cellular Automaton." Journal of Building and Environment. Elsevier. 40(3):411-415.

[17] Goyal, S., H. Ingley, and P. Barooah. 2012. "Zone-level control algorithms based on occupancy information for energy efficient buildings." American Control Conference.

[18] Wang, S., M.V. Schyndel, G. Wainer, V. Subashini, R. Woodbury. 2012. "DEVS-based Building Information Modeling AND Simulation for Emergency Evacuation." In Proceedings of the 2012 Winter Simulation Conference. Berlin, Germany. IEEE.

[19] Hammad, A., and C. Zhang. 2011. "Towards Real-time Simulation of Construction Activities Considering Spatio-temporal Resolution Requirements for Improving Safety and Productivity." In Proceedings of the 2011 WSC. Phoenix, AZ. 3533-3544. IEEE.

[20] Ahmed, A., G. Wainer, and S. Mahmoud. 2010. "Integrating Building Information Modeling & Cell-DEVS Simulation." In Proceedings of SimAUD 2010. Orlando, FL.

[21] IFC2x3. BuildingSmart. 2012. Accessed Nov 6. http://www.buildingsmart-tech.org/ifc/IFC2x3/