Modeling and Simulation of Crowd using Cellular Discrete Event Systems Theory

Ronnie Farrell, Mohammad Moallemi, Sixuan Wang, Wang Xiang, Gabriel Wainer
Dept. of Systems and Computer Engineering
Carleton University, Ottawa, ON, Canada
Centre of Visualization and Simulation (V-SIM)
{ronnie;moallemi;swang;wxiang;gwainer}@sce.carleton.ca

ABSTRACT
In this paper, we discuss how Cellular Discrete Event System Specification (Cell-DEVS) theory can be used in modeling and simulation of the crowd. We will show that the efficient cell update mechanism of Cell-DEVS allows for more efficient entity-based simulation of the crowd compared to cellular automata. On the other hand the formal interfacing mechanisms provided by this theory allows for integration of other components such as DEVS atomic processing component or visualization and building information modeling components with the Cell-DEVS model. Finally, we describe in details of the design and development of several pedestrian models and present the results.

Categories and Subject Descriptors
I.6.5 **[MODEL DEVELOPMENT]**: Modeling methodologies;
I.6.8 **[SIMULATION AND MODELING]**: Types of Simulation –Discrete event, Visual.
J.6 **[COMPUTER-AIDED ENGINEERING]**: Computer-aided design (CAD)

General Terms
Algorithms, Human Factors.

Keywords
Cellular Discrete Event Systems; Pedestrian; Crowd.

1. INTRODUCTION
Human society is a diverse yet cooperative environment in which each member decides and acts independently while the action of the entire members can converge to a single and workforce. This outcome can be constructive or destructive and it depends on various physiological, psychological and social factors. Modeling such kind of behavior can provide insight in learning about human behavior. In particular, Modeling and Simulation (M&S) of crowd behavior can be used for predicting the impact of the behavior of a group of people in different situations. This topic has been of interest in many different fields such as safety, Architectural design, Industrial design, computer games, and transportation.

SIGSIM-PADS'13, May 19–22, 2013, Montréal, Québec, Canada.

Crowd behavior simulation has been studied to investigate possible risks in architectural designs and emergency management systems. These include physics-inspired models such as Helbing's social force model [1], cellular automata models based on predefined rules [2], and agent-based models that use multiple agents to simulate a crowd where each agent has its own behavior and interacts with other nearby agents [3]. Computer-based simulation of the crowd can be categorized based on the size of the model (large, medium, small) and also duration of the simulation (long-term and short-term) [4]. In small-size crowds the modeler can implement the behavior of each individual in the crowd as a separate entity; while in a huge size crowd, the focus is usually on the emergent behavior. From this perspective, crowd modeling approaches are divided to three categories: *flow-based*, *entity-system*, and *agent-based* approaches [4]. Flow-based approaches (e.g. [5]) consider the crowd as a flow of movement that is focused on the overall physical aspects of the crowd with low granularity, which is suitable for large-size simulations due to its light computational demands. Entity-system approaches attack the problem in a more detailed perspective, considering each individual as a separate entity that can render social and physical behavioral aspects. The particles are homogeneous entities that form a crowd in which the movement, speed and social behavior of the individuals are controlled by a set of laws. Hence, this approach demands higher computation and presents more details of the interactions between the particles. Examples of this approach are presented in [6] and [7]. The agent-based approach reflects each individual as an independent and autonomous entity that can behave based on specific rules defined for that entity. The local phenomena can affect the decision-making process of each individual, while the entire crowd can produce an emerging pattern that is deducted by the social and physical aspects of each individual. Due to customized and independent decision-making processes performed by each individual, this approach requires heavy computational resources to be implemented and it is usually not suitable for large-size or long-term simulations. On the other hand, with the rapid advancement of processing resources and sophisticated flexibility and accuracy offered by this approach, it is gaining significant scientific attention. In general, the agent-based models are independent individuals governed by local rules, while entity-based models are homogeneous entities governed by global rules that are semi-independent and reply to local events.

There are different behavioral factors that the modelers capitalize on when modeling crowds. Physical factors consist of position, size, speed, direction, power, etc. Psychological factors involve stress level, courage, etc. Finally, the social factors include family ties, culture, leadership, laws and regulations, etc. Crowd models can be evaluated based on flexibility, extensibility, execution performance, and scalability.

As discussed earlier, M&S has been used as a tool for efficient analysis, design, and verification of crowd behavior. It provides a hierarchical design scheme in which higher abstract levels are branched into levels that contain more details. In particular, the use of a formal M&S methodology would allow one to express groups of people using mathematical notations in which the details of the behavior of the crowd are accurately modeled. Other advantages of formal M&S include applying formal model checking techniques at design time, the incremental refinement of the initial simulation models, simulation-based validation, and reuse of the existing models.

One of the promising theoretical approaches towards resolving crowd modeling specific problems is Cellular automata (CA) theory [10]. In this branch of discrete dynamic systems, which space is represented by a regular grid, with each cell being a state machine. The time advances in a discrete manner, triggering state changes in the cells, based on the value of their neighbor cells. This theory is used in physics, complexity science, theoretical biology, microstructure modeling, and spatial modeling. The Cell-DEVS formalism [9] is derived from both CA and Discrete-EVent System Specification (DEVS) [8]. The Cell-DEVS formalism solves the problem of unnecessary processing burden in quiescent cells, and it allows for a more efficient asynchronous execution.

DEVS is a well-known formal discrete-event M&S methodology, which is based on generic dynamic systems theory. Discrete-event approaches provide event-driven response to external stimuli, which can represent human reaction to the environment. On the other hand, discrete-event methodologies provide continuous timing of the events. In particular, DEVS includes well-defined coupling of components, hierarchical, modular construction, support for discrete event approximation of continuous systems and support for repository reuse. Based on these ideas, Cell-DEVS uses a continuous time base, and provides advanced timing constructions for the cellular models. In this methodology, each cell is represented as a DEVS atomic model that changes state in response to the occurrence of events in an event-driven fashion. Cell-DEVS was originally introduced for modeling and simulation of spatial systems and, as it will be shown in this article, it can be seamlessly adapted for crowd modeling.

We used the DEVS and the Cell-DEVS formalisms for the following reasons:

- DEVS provides a formal foundation to M&S that proved to be successful in different complex applications [9].

- It integrates a simulation approach with the benefits of a formal modeling technique, which provides fast prototyping and incremental development while allowing for reuse of previously existing models.

In the remaining sections of the paper, we discuss the advantages of using Cell-DEVS formalism and show how it can be used to model the crowd in an entity-based approach. The rule-driven nature of cell behavior definition provides a platform for particle-based decision-making definition, leading to easier and faster adoption and implementation of entity-based crowd modeling algorithms. The other advantage of this method is its fast computing apparatus working asynchronously on the cellular grid, increasing the execution speed compared to CA, hence allowing for long-term crowd simulation. Finally, the formal I/O port definitions in the formalism, allows for interfacing variety of environment models with different occupants, and data transfer between different spatial components. The idea is to convert urban environments or architectural spaces into cellular models and provide diverse sceneries to be simulated in 2D and even in 3D cellular models.

2. RELATED WORK

Research on flow-based crowd modeling assumes a crowd of people as a flow of liquid and tries to solve the problem using differential equations. This continuous modeling approach has been investigated in [11], where the authors include a “thinking” factor that is added to the fluid dynamics to assimilate the flow of humans. Another example is the EVACNET4 [12] software, which provides a M&S environment for building evacuation. The tool allows for modeling the buildings and rooms as nodes in a network whose arcs represent the corridors and pass-ways. The modeler assigns capacities to the nodes and arcs, and travel-time to each of these components. Then, the model is computed using a linear programming approach to find the shortest path during the evacuation of the building. A recent application of this software for evacuation M&S in high-rise buildings has been presented in [13].

As mentioned earlier, these approaches do not provide high-resolution details of the behavior of the individuals in the simulation and are somewhat difficult to model. Instead, other research in this area has focused on visualization aspects of the crowd modeling. In [14] flow tiles have been introduced, in order to model the flow of crowd concentrating on the visual and physical aspects of the crowd movement rather than the social and psychological aspects. Therefore, this technique does not render an accurate simulation of the crowd, but it provides a suitable augmented reality representation of the crowd flow to be used in computer game development.

Other existing long-term crowd models can be categorized as flow-based models that tackle the problem from a higher abstract level. Examples of these models are the social life and neighborhood evolution models presented in [15].

There has been a series of models investigating the spread of extremist opinion and beliefs among people in long-term. In [16], the authors investigated the spread of extremism in four different networks of people in which different patterns have emerged based on factors such as uncertainty, influence of other peoples’ words and the learning process of each individual. The uncertainty factor in each individual represents the chances that an individual can give up his/her opinion and accept extremist ideologies. The decision-making of an individual is modeled by a mathematical equation representing the evolution strategy of opinion. It is observed that the single extreme convergence is favored by small shortest paths between all pairs of nodes in the network.

Another agent-based simulation of opinion spread among crowd has been done in [17]. The model is a continuous dynamic opinion spread system where self-organizing agents carry out the simulation based on the principle of meta-contrast. The model considers factors such as network interactions on the evolvement of people’s opinions, and indicates that consensus, polarization or extremism can be produced even without interaction with an extremist agent.

An entity-based model of pedestrian reaction in panic situation has been proposed in [18] in which the factors such a s distance, velocity, mass, and forces are incorporated to quantitatively assess the panic. Each individual is represented as an entity containing mass and power which affects the socio-psychological factors in decision making.

A similar work to ours has been introduced in [19] in which a cellular automata-based small-size model of the environment and the crowd of people as agents is integrated with virtual reality engine to render a 3D representation of the simulation. In this model, spatial representation of the sources of fields that are signals that diffuse in the environment interact with crowd agents with a "perception of signal" factor that is the state of the agent. This perception can lead to the behavioral reactions of the agents in the environment turning the agent to an autonomous or controlled agent. Other factors also can be introduced to the field such as emission-diffusion-perception mechanism that can affect the behavior of the agents.

Another very similar and motivating pedestrian movement M&S approach has been introduced in [20]. In this research, bi-directional pedestrian flow has been modeled using Cellular Automata considering different behavioral factors such as position exchange, lateral-move and step back. As an entity-based model, it defines a Moore neighborhood for each entity and applied sophisticated rules to achieve a realistic pedestrian movement in urban walkways. The position exchange and lateral movement allowed for investigating and increasing the speed of movement in low-density areas and provided a high-resolution simulation of the crowd however, the collision is not managed very well.

Nevertheless, as mentioned earlier the efficient and intelligent computation of cell-state variations in Cell-DEVS (compared to CA) and the interfacing mechanisms provided by this technique allows for developing larger models and faster execution of models as well as straightforward integration of these models with other visualization or modeling tools. We will present different models developed using Cell-DEVS and capitalize on these advantages.

On the other hand, most modeling and simulation methods listed above run on single-user workstations, which normally require too much time for installation and configuration of all the software and dependencies needed for simulation. It is more efficient to access simulation resources remotely by web services, improving scalability (e.g. accessibility, availability and interoperability). In this way, users can reuse and share simulation resources on-site without worrying about the limitations of their local workstations in terms of processing power or memory capacity. SOAP-based Web Services use the Remote Procedure Call (RPC) to allow two distributed heterogeneous systems to call the functions of each other; but it exposes the implementation details and lacks scalability [21]. Another better way is to use RESTful Web Services. Its purpose is to transfer messages of Web resources using a set of uniform operations. Access to RESTful Web Services is through Web resources (URIs) and XML messages using HTTP methods (GET, PUT, POST and DELETE). Based on these ideas proposed in [21], the authors presented the RESTful Interoperability Simulation Environment (RISE) middleware. The main objective of RISE is to support interoperability and mash-ups of distributed simulations regardless of the modeling formalism, programming language, or simulation engine.

3. Crowd Modeling using Cell-DEVS

Cell-DEVS [9] is an extension to DEVS [8] that allows defining cellular models with explicit timing delays. A Cell-DEVS model is a lattice of cells holding state variables and a computing apparatus, which is in charge of updating the cell states according to a local rule. This is done using the current cell state and those of a finite set of nearby cells (called its neighborhood). Cell-DEVS improves execution performance of cellular models by using a discrete-event approach. It also enhances the cell's timing definition by making it more expressive. Each cell is defined as a DEVS atomic model, and it can be later integrated to a coupled model representing the cell space. Cell-DEVS models are informally defined as shown in Figure 1.

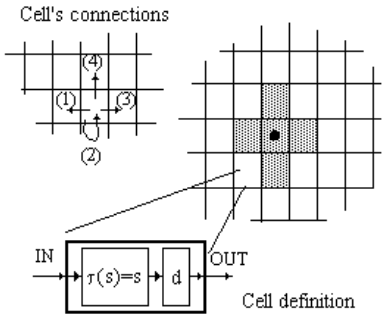


Figure 1. Cell-DEVS model [9].

CD++ [9] is a M&S tool that provides a development environment for implementing DEVS and Cell-DEVS models. DEVS atomic models can be developed and incorporated into a class hierarchy programmed in C++. Coupled models can be defined using a built-in specification language. Cell-DEVS models are built following the formal specifications for DEVS, and a built-in language is provided to describe the behavior rules. The language is based on the formal specifications of Cell-DEVS. The model specification includes the definition of the size and dimension of the cell space, the shape of the neighborhood and borders. The cell's local computing function is defined using a set of rules with the form POSTCONDITION DELAY {PRE-CONDITION}. These indicate that when the PRECONDITION is satisfied, the state of the cell will change to the designated POSTCONDITION, whose computed values will be transmitted to other components after consuming the DELAY. If the precondition is false, the next rule in the list is evaluated until a rule is satisfied or there are no more rules.

3.1 Building Information Modeling

In our previous research presented in [22], Cell-DEVS has been used in modeling evacuation of the crowd and detecting bottlenecks and congestion points in the buildings. Cell-DEVS models developed in CD++ were interfaced with Building Information Modeling (BIM) [23] models developed in Autodesk Revit™ Architecture [24] in order to export the building information from BIM and feed it to Cell-DEVS model to compute the crowd evacuation procedure. The results of the simulation in the Cell-DEVS model are then transported back to Autodesk 3D Max™ [25] to visualize of the simulation steps.

Figure 2 illustrates the three components used in the simulation of the crowd evacuation, where the BIM model data are extracted from Autodesk Revit™ tool using the APIs and the BIM add-in, and stored in CD++ initial value file. This file contains the initial

coordinates of the BIM elements extracted from BIM model by the Parameter Extractor component from *Coordinate Loader*. Some other rule-construction data are also extracted and stored in CD++ *include* files. After this, a 2D cellular model of the BIM system is constructed in CD++ based on the dimensions and initial values extracted from BIM model. The multi-level cellular evacuation model defined in CD++ is declared using a set of rules indicating the output value for the cell's state after satisfying a precondition. A neighborhood consisting of down, left, up and then right cells that are adjacent to the central cell is declared. The model performs the following five rules wherever they apply, in order to simulate the evacuation in the building: 1) someone enters the current cell; 2) someone moves from the current cell to another one, 3) someone exits the floor, 4) someone enters a stairwell, and 5) someone exits a stairwell. After the Cell-DEVS model is executed in CD++ and the results are stored in a log file, they are then imported to the 3D Max™ to be visualized in 3D. Finally, the produced log file is parsed by looking for simulation times, cells locations and values and then the simulation scenario is visualized in the already created 3D visualization of the initial BIM model, imported from Revit™ to 3D Max™ by the *Data Loader*.

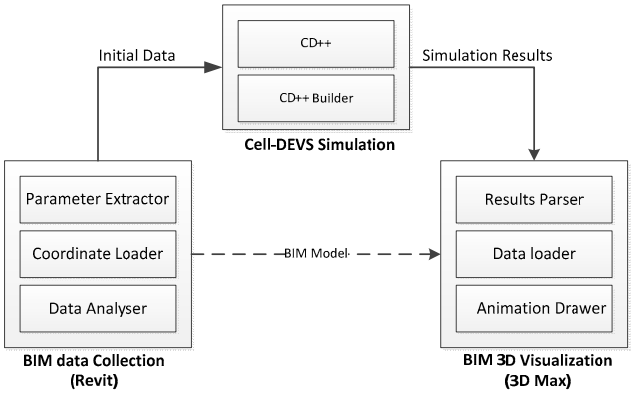


Figure 2. Interactive simulation system architecture [22].

3.2 Bi-directional Pedestrian Flow

Pedestrian movement M&S tries to render the intelligent, physical, and social behavior of the people in which the resultant movement of the crowd in different situations is calculated based on predefined rules. We modeled three different situation 1) Pedestrian walkway 2) Advanced pedestrian walkway 3) Pedestrian movement at doorway.

3.2.1 Pedestrian Movement on Walkway

Figure 3 shows the DEVS model definition diagram of this project. The *Pedestrian Gen* blocks are atomic DEVS components producing random sequence. The *Walkway* is a 2D Cell-DEVS coupled component of size 8×20 cells with the top-most and the bottom-most rows representing the walls of the walkway. Pedestrians are generated at the left and right ends of the walkway. The DEVS random generators are interfaced with the Cell-DEVS *Walkway* coupled component through the I/O definitions provided by DEVS formal structure. The goal of the project is to program the "pedestrians" to travel to the opposite end of the walkway without collision. The speeds of all "pedestrians" are modeled at a constant rate, and everyone travels at that same speed.

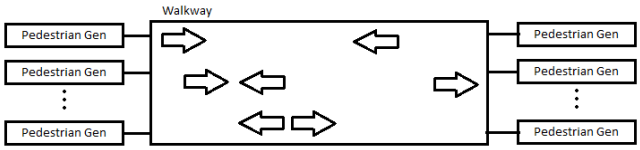


Figure 3. Bi-direction pedestrian Cell-DEVS model.

The rules that are used to regulate the behaviors of the pedestrians are explained below in detail.

Rule 1: Moving forward rule: If the cell in front of a "pedestrian" cell from either direction is empty, and no other "pedestrian" from the opposite direction is located two cells ahead, move forward. This is the highest priority rule. No action from other rules can take the empty square of moving forward rule (See Figure 4).

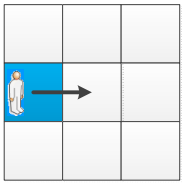
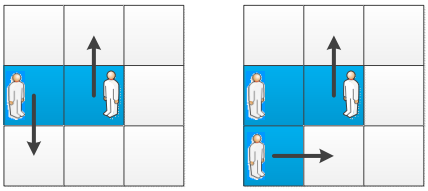


Figure 4. Rule 1.

Rule 2: Side stepping when straight movement is blocked: If a "pedestrian" is blocked by the opposite side "pedestrian", both "pedestrians" try to step to the right if that cell is empty (Figure 5.a). If the "pedestrian" is blocked by another one traveling to the same direction, simply do nothing but wait the other one to move first (Figure 5.b). This is the second highest priority rule.



a) b)

Figure 5. Rule 2.

Rule 3: Side stepping when trying to take the same cell for straight movement: If two "pedestrians" from opposite side trying to take the same empty cell, both step to the right (Figure 6). If they are travelling in the same direction, Rule 1 takes care of it. This is the third highest priority rule.

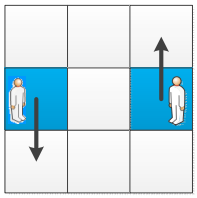


Figure 6. Rule 3.

Rule 4: Side step to the left if the right side is a wall: For the cases where side step to the right blocked by a wall (borders of the model), the "pedestrians" try to step to the left if there is an empty cell. This rule applies to both side-stepping when blocked or trying to take the same cell (Figure 7).

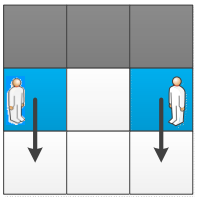


Figure 7. Rule 4.

These rules are a subset or a generalization of the 8 rules provided in [20] according to human natural way of walking. Moving straight forward is the most fundamental and most important form of traveling. Pedestrians change lane only when necessary. Also, people tend to step to the right when encountering another pedestrian traveling the opposite direction. Given everyone travel at same speed, it does not make sense to make a side step when encounter someone with the same direction as the pedestrian. Aside from these basic rules, other rules to control the entrance and exiting of the pedestrians have been implemented.

A customized neighborhood is defined for the cells based on the aforementioned rules. Cell (0,0) represents the core cell which is the pedestrian and it can move to the adjacent cell either north, south, east or west based on the rules specified above. In this neighborhood definition, each cell must watch 2 cells ahead of itself. Because cells move to either direction, therefore we need to include 2 cells from each side. Each cell also needs to watch 2 cells on its right and left side.

		(-2,0)		
(-1,-2)	(-1,-1)	(-1,0)	(-1,1)	(-1,2)
(0,-2)	(0,-1)	(0,0)	(0,1)	(0,2)
(1,-2)	(1,-1)	(1,0)	(1,1)	(1,2)
		(2,0)		

Figure 8. Neighborhood for pedestrian walkway.

This model has been implemented on CD++ with the following assumptions:

Assign a value of 2 to the pedestrians who are walking towards left. Assign a value of 1 to the pedestrians who are walking towards. Empty cells have a value of 0 and walls are 3. If cells with value of 2 attempt to step to the cell above and found out the above row is the "wall", the cell then try to move to the cell below. The opposite rule applies to the cells with value of 1.

The results of this model were visualized using CD++ Animation tool. The blue cells are pedestrians moving to right direction and the red cells move to left. Black cells are walls. In Figure 9 and Figure 10 the side stepping event is extracted from the animation where in left the first step shows that cells that are blocked turn right to avoid collision. In Figure 9 the red cell on the top turns left to avoid the oncoming blue cell and the wall on the right.

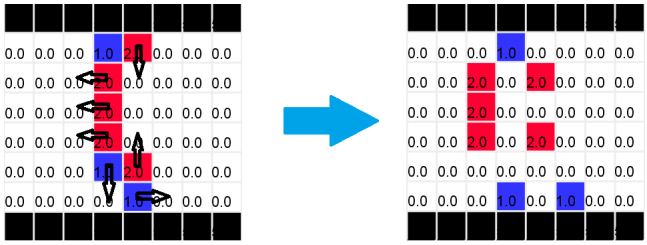


Figure 9. Side stepping rule when blocked 1.

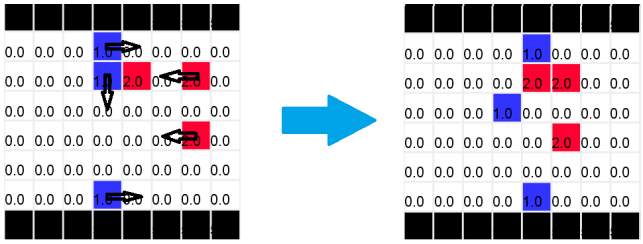


Figure 10. Side stepping rule when blocked 2.

The indicated cells in Figure 11 try to avoid entering the same cell, thus the blue cell turns right. However the red cell freezes in this step because its right neighboring cell is already occupied. In Figure 12, the indicated cells avoid entering the same cell thus the blue cell turns right while the red cell turns left because its right neighbor is a wall. Therefore, they encounter the same situation in the next step, however this time both turn right and continue their movement.

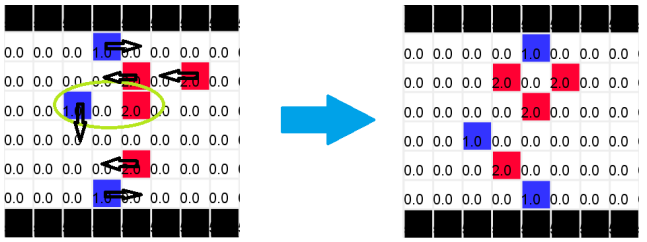


Figure 11. Side stepping due to heading to the same cell.

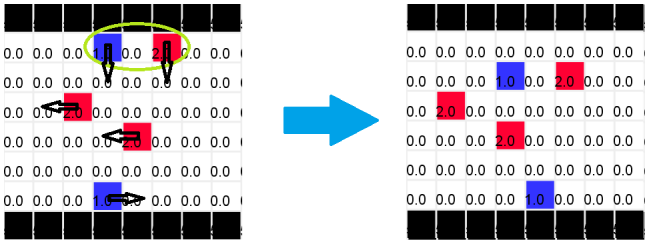


Figure 12. Boarder behavior.

3.2.2 Advanced pedestrian walkway

Observed from the previous approach, the priority scheme is basically an order of sequence in which the rules should be executed in TIME. The basic idea is that after higher priority rules made the changes, the lower priority rules only need to check if the cell it attempts to move to is empty. This is because all other higher level priority rules have already been executed, and thus the only empty cells left are for the current level of priority rule to use. After many implementation experiments performed, the following is the final design of this approach. In this approach we aim at avoiding any collision which was not supported in the previous model.

The fundamental conflict of sidestepping is due to two cells try to sidestep into the same empty cell from the opposite side. One of the cells has to step upwards, while the other one has to step downwards to result in the conflict. Then instead of applying the priority scheme to the time of execution, apply priorities with respect to the directions of pedestrian movements. For example, the highest priority is assigned to horizontal movements or the straight movements from both directions. For this level of priority, Cell-DEVS only needs to check if the cell they are going to move in is empty, and there is no other cell attempting to move into the same cell from the other direction. Then assign the second priority to the rules that would result in sidesteps upwards. These rules includes sidestepping to the right if the pedestrian is traveling to the left; also includes sidestepping to the left if the pedestrian is travelling to the right and there is a wall to the right of the pedestrian. This level of priority rules not only need to check if they need to perform a sidestep possibly due to blocked by another cell, they also need to check if level 1 priority rules can happen to the cell they attempt to move to. Lastly, the lowest priority is assigned to the rules that would result in sidesteps downwards. These rules basically are composed of the opposite rules as priority 2 rule set. However, this lowest priority rule set need to check all higher priority rules. If there is another cell that can step upwards into the target cell, then the testing cell cannot step downwards to that same cell. It seems the check procedure grows exponentially towards lower priority levels. However, there are only 3 levels in the system, so this design is promising.

With the new priority scheme developed, the implementation of the scheme was surprisingly simple and successful. No 3D cell space is needed, nor multiple state variable are required. Only changes required to make was to regroup the rules and develop a few more precondition checks for level 3 priority rules. With this approach, there are only 5 more cells added to the neighborhood (see Figure 13) comparing to the implementation of the previous model. This is an acceptable sacrifice to achieve conflict free simulation.

		(-2,0)		
(-1,-2)	(-1,-1)	(-1,0)	(-1,1)	(-1,2)
(0,-2)	(0,-1)	(0,0)	(0,1)	(0,2)
(1,-2)	(1,-1)	(1,0)	(1,1)	(1,2)
(2,-2)	(2,-1)	(2,0)	(2,1)	(2,2)
		(3,0)		

Figure 13. Neighborhood for advanced pedestrian walkway.

Figure 14 shows the new behavior after the implementation of the sidestep checks. As shown in the green circle, the cell with value of 1 is blocked for moving straight to the right; and the cell with value of 2 is blocked for moving straight to the left. This is the situation where the conflict of sidestepping would occur. In the implementation of the previous model, the cell with value of 1 would step downwards, and the cells with a value of 2 would step upwards. Thus, one cell will overlap another and thus "kill" one pedestrian. Now, after the implementation of sidestep check with priority scheme, this conflict does not occur anymore. After those two cells in the green circle discover that they are blocked towards moving straight forward, the second level priority rule tells the cell with value of 2 to check if the cell above is empty. If it is empty, step upward. Then the third level priority rule tells the cell with value of 1 to check whether the cell below is empty or there is another cell moving into it. Since there is indeed a cell moving into the empty cell, the cell with value of 1 waits in this cycle. One more thing worth to mention is that the priority scheme makes the pedestrian flow very naturally. Imagine that there was no priority scheme applied to the model, but all rules need to check all other rules. Those two cells in the green circle will both detect that the other one is potentially moving into the empty cell. Thus both cells will just wait in that cycle. This is not very desirable and might cause a traffic jam in some situation. Figure 14 does not really illustrate this situation, but it is definitely desirable to have one cell to move to ease the traffic.

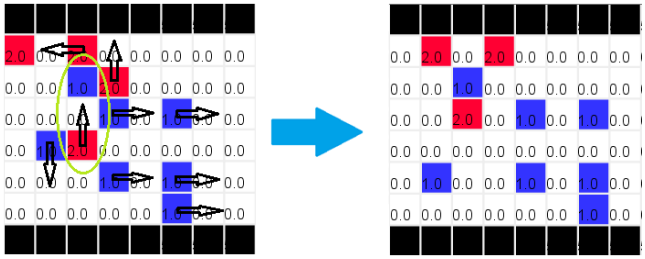


Figure 14. Conflicting sidestep rules.

3.2.3 Pedestrian movement at doorway

A new simulation model using the pedestrian rules from the previous models is proposed that involves taking two room components and allowing pedestrians to walk between them through a doorway. The purpose of this project is to simulate the effect that pedestrians have while walking in a doorway. In order to cover all cases of pedestrian behavior, new rules will need to be implemented, aside from the existing rules. This new model involves creating two different Cell-DEVS components; one to represent the room, and the other to represent the hall. These two components both use the pedestrian behavior rules mentioned earlier. In order to get pedestrians to walk from one component to the other, a DEVS model is created and interfaced to both cell-DEVS models, which connect the two components together. In addition, in order to obtain proper behavior of the pedestrians when they reach the door, a new zone was implemented.

Figure 15 shows the coupled model definition of the project. The *Pedestrian Gen* blocks are DEVS atomic random sequence generator components. The *Hall* is a Cell-DEVS model with a cell space of 10×6 cells. The *Room* is also a Cell-DEVS model with a cell space of 10×10 cells. The *Door Controller* block is a DEVS atomic component that has two input ports, and two output ports (one for the hall and one for the room).

When the Door Controller receives a message on an input port, it sends the value of the message to the associated output port after a specific time. For example, when port *inHall* receives a value of 2, port *outHall* will generate a value of 2 after 50msec. In this model there are two *Door Controller* components representing a doorway with two doors.

The doorway which connects the room and hall components is represented by dotted lines in the coupled model. It consists of cells (0,2) and (0,3) in the *Hall*, and cells (9,4) and (9,5) in the *Room*. The doorway cells have both input and output ports associated with them, as pedestrians can both enter and exit the areas through the door. The output ports of the hall and room components are connected to the input ports of the *Door Controller* component respectively. The output ports of the *Door Controller* component is connected to the input ports of the *Hall* and *Room* components respectively.

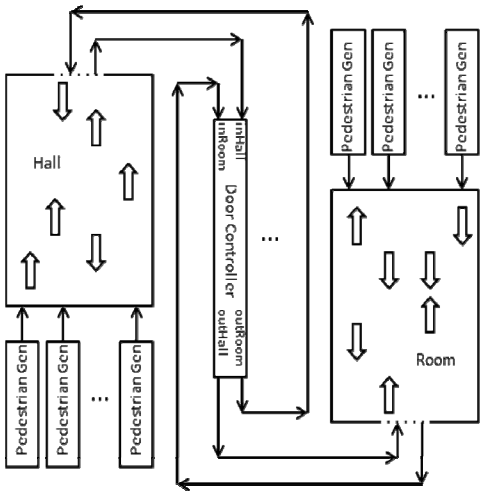


Figure 15. Cell-DEVS Coupled Model.

Pedestrians are generated at the bottom of the hall, and at the top of the room. Once the pedestrians reach the doorway, they leave their current area through the door, and enter the other area through the door. The goal of this project is to program the pedestrians to walk from one room to the other via the door, and see the effects that result from the doorway. The speed of all the pedestrians are modeled as a constant speed and everyone travels at this rate. To do this, some rules are used to regulate the behaviors of the pedestrians.

Figure 16 is a graphical representation of the neighborhood. The neighborhood has been increased to avoid collisions.

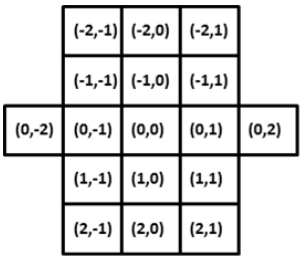


Figure 16. Neighborhood for pedestrian doorway.

As mentioned earlier, to manage the pedestrian movement at the boundary of the *Room* or the *Hall* components, extra rules are needed. In order to have the pedestrians walk towards the door (and not to step to the right to avoid a collision) a new zone was implemented at the boundary. The followings are the rules for pedestrians at the boundary of the *Hall*. The same logic applies to the rules for the *Room*.

Rule 1: Up walker at the door, exit Hall: The up walker enters one of the door cells (i.e. value of 1 on cell (0,2) or (0,3)). To represent the walker exiting the *Hall*, in the next time step the cell will become empty. This rule also sends a value of 1 to the doors output port which is connected to the input port of the Door Controller DEVS model. This rule has the highest priority.

Rule 2: Enter boundary: There is an up walker at the boundary, have them walk forward one cell into the boundary in the next time-step, if the intended cell in the boundary is empty. This rule has the second highest priority.

Rule 3: Up walker in boundary to the right of the Door walk left if possible: There is a walker in the boundary, and they are on the right side of the Door, move to the left adjacent cell if that cell is unoccupied, and another walker is not going to take that cell in the next time-step. This rule has the third highest priority.

Rule 4: Up walker in boundary to the left of the Door, walk right if possible: There is a walker in the boundary, and they are on the left side of the door, move to the right adjacent cell if that cell is unoccupied, and another walker is not going to take that cell in the next time-step. This rule has the fourth highest priority.

Rule 5: Down walker that entered Hall from Door, leave boundary: A down walker that has entered the hall boundary from the room through the door, they leave the boundary in the next time step if the adjacent cell below them is empty.

Rule 6: Default: If none of the other rules execute, the walker does nothing and waits where they are. This rule also contains the behavior for the obstacles, as they never move.

After implementing the model on CD++, we show some of the results generated by CD++ Animation tool that are related to the door entrance and exit, to emphasize on the new features of this model. Figure 17 demonstrates that the up walkers are in fact able to enter the boundary of the hall, and that down walkers are able to enter the boundary of the room. The boundary is represented by a blue rectangle. The larger rectangle is the boundary of the room, whereas the smaller rectangle is the boundary of the hall. It also demonstrates how the walkers leave the boundary.

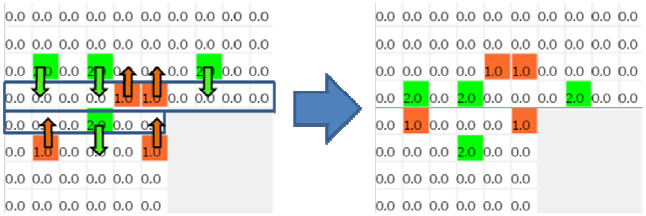


Figure 17. Entering Boundary.

Figure 18 demonstrates how pedestrians in the hall, who are trying to walk to the room (orange), walk towards the door (red boxes) in the hall. The same is true for the pedestrians in the room trying to walk to the hall. More videos of this model can be watched here: http://www.youtube.com/watch?v=I77-zLCRpS4

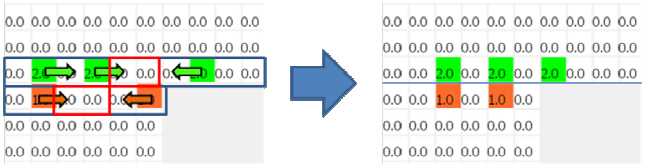


Figure 18. Move towards door.

4. INTEGRATED FRAMEWORK

After modeling the details of pedestrian movement, we are interested in integrating this cellular simulation framework with Computer-Aided Design (CAD) tools. Figure 19 illustrates the integrated Modeling, Simulation and Visualization Framework. The overall approach includes four subsystems: Data Collection, Cell-DEVS modeling, Remote Simulation and 3D Visualization. This framework relies on the CAD file/databases supported by different CAD tools as an information repository for different

transportation facilities projects. These CAD files are object-oriented, attribute-driven with 3D geometric information.

The main four steps in the integrated framework are:

1) Data Collection: it is done automatically, generating initial data files from CAD files into the Cell-DEVS model. This includes an *Element Filter* for selective properties, and a *Coordinate Extractor* for getting *(x, y, z)* coordinates of filtered elements. The generated files are used for pedestrian movement model inputs explained earlier.

2) Cell-DEVS Modeling: it builds a Cell-DEVS pedestrian movement model according to the collected data and models the pedestrian behaviors.

3) DEVS Simulation: it submits Cell-DEVS model to RISE Web Services URI, executes the simulation remotely, and then gets the simulation results.

4) BIM 3D Visualization: it visualizes the simulation results in 3D, based on transportation facilities CAD file, providing an intuitive mean for analysis. It parses the simulation log files, and loads it using *GUI Script*. The results are generated by the *Models Animation*.

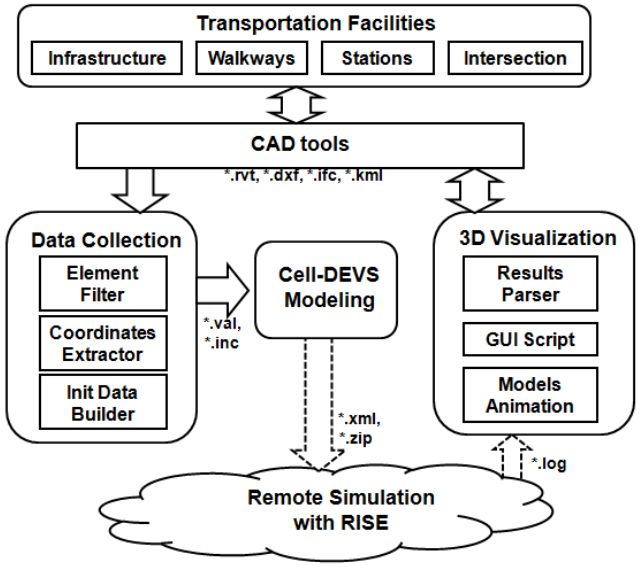


Figure 19. Integrated Modeling, Simulation and Visualization Framework.

4.1 CAD Data Collection

With the development of digital technology, CAD has gained with excellent capability to support management of graphical and geometric information. Such information has been used in object-oriented modeling and building information modeling (BIM) to evaluation of the designs. Therefore, pedestrian movement analysis can benefit from CAD tools from which information of transportation facility can be extracted. In current implementation, we chose Autodesk Revit Architecture for this purpose, as it includes a well-defined parametric API and a BIM add-in. Revit can be used to build 3D models and 2D drawings, and the Revit API can be used to find and manipulate different model elements (walls, doors, obstacles, etc.).

As seen from Figure 19, for getting information from CAD for initial date to our Cell-DEVS model, we need three main steps: 1) Filtering elements; 2) Extracting Coordinates; 3) Building Initial files.

For the first step, we use Revit build-in API to filter needed element type (e.g. wall, obstacles), then extract parameter properties that we need (e.g. length, width). For the second step, based on the filtered elements ID, we extract their coordinates according to the coordinates system. Figure 20 shows a segment after these two phases, in which each line represents wall (Element ID, coordinates, width, length and area scale).

--Filter Elements & Extract Coordinates
180921: x(-524") y(314") z(0") width(8") length(600") area(480.00 SF)
180922: x(24") y(314") z(0") width(4") length(600") area(240.00 SF)
180923: x(-524") y(914") z(0") width(8") length(600") area(480.00 SF)
180924: x(24") y(914") z(0") width(4") length(600") area(240.00 SF)

Figure 20. Result segment of Data Collection.

For the third step, based on the extracted data, we can generate initialization data files for the Cell-DEVS model. For example, in this occupancy models, we firstly calculate the scale of the space, converting it to a unified size: *XMIN* (the horizontal ordinate of the left-most point) and *XMAX* (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 the three dimensions, and next generate initial **.val* file which contain initial value of the grid. The basic idea of initializing state of each cell is to "pave" all needed elements according to the extracted info (starting points, width, length, and so on). For the pedestrian model, we got (30×30) size of space and the related initial **.val* file.

4.2 RISE Remote Simulation

As discussed in section 2, RESTful Web Services can further increase the scalability in terms of the plug-and-play interoperability and availability. In [21], the authors presented the first existing RESTful Interoperability Simulation Environment (RISE) middleware. RISE is accessed through Web resources (URIs) and XML messages using HTTP methods. Implementations can be hidden in resources, which are represented only via URIs. Users can run multiple instances as needed, which are persistent and repeatable by specific URIs. The HTTP methods are typical four types: *GET* (to read a resource), *PUT* (to create or update a resource), *POST* (to append data to a resource), and *DELETE* (to remove a resource). An interface between RISE and CD++ (supporting DEVS and Cell-DEVS) allows running DEVS/Cell-DEVS distributed simulations. RISE API likes URL http://www.example.com/cdpp/sim/workspaces, attached by the following services:

- ***../{userworkspace}*** contains of all simulation services for a given user.
- ***../{userworkspace}/{servicetype}*** contains of all frameworks for a given user and simulation service type (e.g. a simulation engine CD++).
- ***../{userworkspace}/{servicetype}/{framework}*** allows interacting with a framework (including the simulation's initial files, configuration and src code).

- ***../{userworkspace}/{servicetype}{framework}/simulation*** interacts with the simulator execution.
- ***../{userworkspace}/{servicetype}/{framework}/results*** contains of the simulation outputs.
- ***../{userworkspace}/{servicetype}/{framework}/debug*** contains of the model-debugging files.

For a specific simulation, we can realize the remote simulation by using these URIs with HTTP methods. For example, after getting initial model and configuration files for simulation, we use *PUT* to create a framework with the configuration file and *POST* to upload these initial model files to this framework. Then, this newly created simulation environment can be executed by using PUT to *{framework}/simulation*, then we wait for the simulation to finish and *GET* the simulation results files from *{framework}/results*.

4.3 3D Visualization

3D visualization provides a more intuitive and attractive way to obtain visual simulation results in CAD tools, enabling the designers to check the building performance and people behaviors under different properties. Most CAD tools support full-featured 3D visualization of building. Among them, Autodesk 3ds Max is a powerful option for its animation and rendering ability for 3D visualization. In [26], 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), expanded GUI in 3ds Max with extendable script. The GUI 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, find the flaws and plan for improvement. Figure **21** shows its UML class diagram, which is based on the Model-View-Controller design pattern. The controller is *CDVizController*. The simulation grid is represented by an array of *GridCell* objects, which keep track of which person is in that cell, and each person in the simulation is represented by an instance of *PersonModel*.

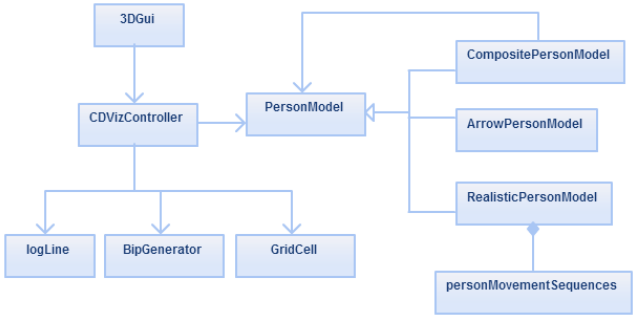


Figure 21. Class diagram of the visualization plug-in [26].

5. CASE STUDY

Now, let us see how to use the integrated framework proposed in this paper to increase scalability and interoperability. Firstly, we use the developed CAD data collection tool in Revit to get the layout of the studying area of bi-directional pedestrian walkway. We modeled a 30×30 cellular space and the related initial *pedestrian.val* file for walls and obstacles. Then, we upload the initialized files with our Cell-DEVS model (*pedestrian.ma*) for remote simulation. With the RISE server, we can use the PUT method to create a framework *.../PedestrianMovement* for simulation, and the POST method to upload the initial model files. We can run this simulation by using PUT to *.../PedestrianMovement/simulation*. We wait until the simulation finishes, then by using GET method simulation results are fetched from *.../PedestrianMovement/results*. The configuration XML file for this *.../PedestrianMovement* framework in RISE looks like the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ConfigFramework>
  <Doc>This model Simulates bi-directional pedestrian movement using Cell-Devs.</Doc>
    <Files>
      <File ftype="sup">pedestrian.val</File>
      <File ftype="inc">pedestrian.inc</File>
      <File ftype="ma">pedestrian.ma</File>
    </Files>
  <Options>
    <TimeOp>00:00:10:000</TimeOp>
    <ParsingOp>false</ParsingOp>
  </Options>
  <DCDpp>
    <Servers>
      <Server IP="134.117.53.66" PORT="8080">
          <ZONE>pedestrian(0,0)..(29,29)</ZONE>
      </Server>
    </Servers>
  </DCDpp>
</ConfigFramework>
```

Finally, after parsing the simulation log file for the specific 3D visualization, we can load the same version of the CAD file in 3d MAX and visualize the simulation animations with the help of its GUI. Figure **22** shows how the 3D visualization would look like. This tool helps the designers to observe different animation features of pedestrians: the realistic model for real pedestrian movement (a, b and c) and the arrow model for pointing the moving direction (d). According to the performance (occupancy level or bottlenecks analysis) observed from this 3D visualization, all steps can be reused for the next design iteration.

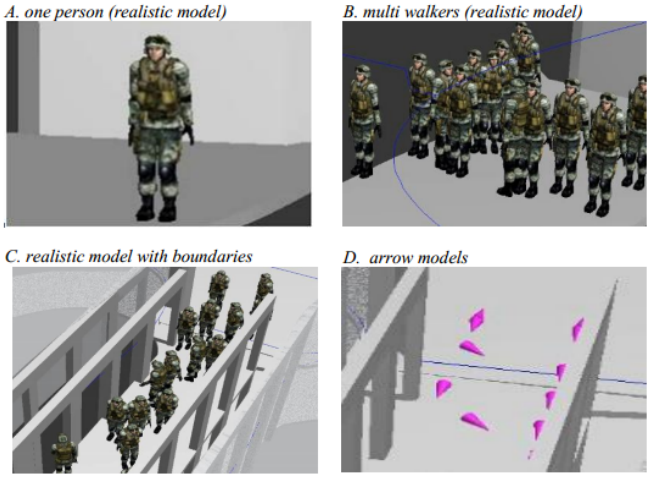


Figure 22. 3D Visualization Results.

6. CONCLUSIONS

We showed how Cell-DEVS can be used in M&S of the crowd by using its cellular structure to render crowd entities, obstacle, or empty space. Multiple simulations were performed, and different techniques were discussed. We showed how Cell-DEVS can be integrated with BIM tools to model building evacuation simulation, thanks to the formal structure of DEVS. In the next series of models, we demonstrated pedestrian movement flow M&S and collisions management. Multiple simulations were also performed while varying the pedestrian's generator values. Based on the experiments it was concluded that Cell-DEVS is very suitable for entity-based simulation of the crowd, capable of simulating small to medium size models. However, large-size models can also be simulated using the flattened coordinator version of the CD++ tool in which the initial cell creation processing is dramatically faster. We showed how to extract information from CAD tool, run simulation remotely on RISE server using Cell-DEVS and view advanced 3D visualization in 3D Max. This system allows designers to analyze the pedestrian movement behaviors more comprehensively, check the performance of the design, and plan for improvements.

7. REFERENCES

[1] Helbing D., Farkás I.J., Molnár P., Vicsek T. 2002 Simulation of pedestrian crowds in normal and evacuation situations, in: M. Schreckenberg, S.D. Sharma (Eds.), Pedestrian and Evacuation Dynamics, *Springer, Berlin,* (2002), 21–58.

[2] Blue V.J., Adler J.L. 2001 Cellular automata microsimulation for modeling bi-directional pedestrian walkways, *Transportation Research Part B 35* (3) (2001) 293–312.

[3] Penn A., Turner A., Space syntax based agent simulation, in: M. Schreckenberg, S.D. Sharma (Eds.), Pedestrian and Evacuation Dynamics*, Springer, Berlin,* (2002), 99–114.

[4] Zhou, S., Chen, D., Cai, W., Luo, L., Low, M. Y. H., Tian, F., ... & Hamilton, B. D. (2010). Crowd modeling and simulation technologies. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(4), 20.

[5] Hughes, R. L. (2003). The flow of human crowds. *Annual Review on Fluid Mechanics* 35, 169–182.

[6] Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Letters to Nature* 407, 487–490.

[7] Castonguay, P., & Wainer, G. (2009, March). Aircraft evacuation DEVS implementation & visualization. *In Proceedings of the 2009 Spring Simulation Multiconference. San Diego, CA.* 144.

[8] Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000) Theory of modeling and simulation. 2000.

[9] Wainer, G. A. (2009). Discrete-event modeling and simulation: a practitioner's approach. CRC.

[10] Wolfram, S. (1986). Theory and applications of cellular automata.

[11] Hughes, R. L. (2003). The flow of human crowds. *Annual Review on Fluid Mechanics* 35, 169–182.

[12] Kisko, T. M., Francis, R. L., & Nobel, C. R. (1998). EVACNET4 User's Guide. *University of Florida*.

[13] Zhang, W. M., Huang, L., & Wang, B. (2009). Application of the EVACNET4 model to evacuation in high-rise building [J]. *Fire Science and Technology*, 3, 014.

[14] Chenney, S. (2004). Flow tiles*. In Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 233-242). Eurographics Association.

[15] Miller, J. H., & Page, S. E. (2007). Complex adaptive systems: An introduction to computational models of social life. *Princeton University Press*.

[16] Deffuant, G. (2006). Comparing extremism propagation patterns in continuous opinion models. *Journal of Artificial Societies and Social Simulation*, 9(3).

[17] Salzarulo, L. (2006). A continuous opinion dynamics model based on the principle of meta-contrast*. Journal of Artificial Societies and Social Simulation*, 9(1).

[18] Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407(6803), 487-490.

[19] Bandini, S., Manzoni, S., & Vizzari, G. (2006). Crowd Modeling and Simulation. *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*, 105-120.

[20] Tao, W., & Jun, C. (2009). An Improved Cellular Automaton Model for Urban Walkway Bi-directional Pedestrian Flow. *In Measuring Technology and Mechatronics Automation, 2009. ICMTMA'09. International Conference* Vol. 3, pp. 458-461.

[21] Al-Zoubi, K. and Wainer, G. (2010). Distributed Simulation Using Restful Interoperability Simulation Environment (RISE) Middleware. *Intelligence-Based Systems Engineering*, Pages 129–157.

[22] Wang, S., Van Schyndel, M., Wainer, G., Rajus, V. S., & Woodbury, R. (2012). DEVS-based building information modeling and simulation for emergency evacuation. *In Proceedings of the Winter Simulation Conference* (p. 60). Winter Simulation Conference.

[23] Wright, F. L. (2009). Building Information Modeling.

[24] AutoDesk. (2013). "Autodesk Revit Architecture." Accessed Feb. 15. http://usa.autodesk.com/revit-architecture/.

[25] AutoDesk. (2013). "Autodesk 3ds Max." Accessed Feb. 15. http://usa.autodesk.com/3ds-max/.

[26] Freire, V., Wang, S., and Wainer, G. (2013). Visualization In 3ds Max For Cell-DEVS Models Based On Moving Entities. *Symposium on Simulation for Architecture and Urban Design (SimAUD'13)*. San Diego, USA.