Egress Modeling with Cellular Discrete Event System

Ryan Lawler
Electrical, Computer, Software and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, FL
lawlerr@my.erau.edu

Shafagh Jafer
Electrical, Computer, Software and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, Florida
jafers@erau.edu

***Abstract*— The Discrete-Event Modeling and Simulation (DEVS) formalism has been successfully used to model systems whose behavioral state changes based on trigger events. The purpose of this project is to explore the extension of DEVS beyond the modeling of event driven systems, focusing on the application of DEVS to changes in human behavior as driven by emergency events. The objective is to build a modeling and simulation framework for human behavior in emergencies that could easily be extended and built upon in the future. Here we introduce the development of a suite of Cell-DEVS models representing aspects of human behaviors, and the simulation and a detailed analysis of this suite of models. This suite of models explored random and controlled human behaviors, as well as implementing psychological conditions such as herd following. The end state is the development of a repository for these models to serve the DEVS community.**

Keywords— Evacuation; egress; emergency modeling; crowd simulation; Cell-DEVS.

I. Introduction

Modeling and simulation of human behavior during emergency situations is not a new concept, with existing efforts having already helped to improve safety, and aid in the development of evacuation methodologies. But, this application of modeling and simulation is far from exhaustive, and there exist much room to improve. In his doctorate dissertation "Computational Modeling of Human and Social Behaviors for Emergency Egress Analysis", Pan, demonstrates that defining human behavior in emergency situations is not a simple activity; emergency situations trigger dynamic changes in human behaviors that can be difficult to model due to perceived unpredictability[1]. A flexible modeling and simulation construct would be effective in this situation, and this is where the DEVS modeling formalism comes into play.

While this paper will be primarily focused on improving the modelling and simulation of human behavior in emergency situations, there is an opportunity to help address an issue within the DEVS community. In his textbook "Discrete-event modeling and simulation : a practitioner's approach"[2], Wainer identified a key issue that with the substantial amounts of literature documenting various theories and methods on discrete-event modeling and simulation, there should be more research and literature aimed at the translation of these theories and methods into practical applications [2]. This paper will contribute towards the development of the DEVS community by providing literature and a suite of models that demonstrate the link between the theory and practical application of DEVS in the context of human behavior during emergency situations.

The work presented here will not attempt to define the human behaviors that emerge during emergency situations, but rather will survey the existing human behavior literature in order to develop a set of behaviors that can then be modelled and simulated using Cellular DEVS (Cell-DEVS) [2] and CD++ [3]. This paper will present some of our efforts in developing a suite of various crowd emergency behaviors in a variety of situations because the set of behaviors people exhibit in emergencies is dependent on a variety of internal and external variables that include, but are not limited to, the type of emergency situation, the amount of people who are impacted by the emergency situation, the amount of people in close proximity, the type of environment and surroundings people are in, as well as the accessibility of evacuation information.

Here we present the development of five models and simulations that can aid in the development of crowd guidance optimization methods in emergency situations. The activities, analyses and results from any specific application of these models are open-source and available to the DEVS community at [8].

The paper is organized as following: Background section will provide an overview of crowd modeling and provide a brief literature survey over the topic. Emergency Evacuation Modeling will present common egress patterns. The discrete-event approach adopted by this paper is explained in Cellular Discrete Event Approach section. The various egress models and their Cell-DEVS details are presented in Proof of Concept and Analysis of Basic Models. The Results section will discuss the statistical analysis of various simulation runs over the developed egress models. Finally, the Conclusion section will summarize the work and provide some future insights.

II. Background

This section provides a brief literature review of previous research targeting the analysis of human behavior in emergencies. Study of this literature provided a roadmap for how to decompose a system for the development of models and simulations, and provided insights on how human behavior in an emergency can be broken down into elements resembling system components with behaviors driven by discrete events.

ESCAPES [4] provides an evacuation simulation tool designed to capture the intricacies when trying to evacuate

978-1-4673-7300-5/15/$31.00 ©2015 IEEE

families from unfamiliar public spaces. In [4], the ESCAPES tool is demonstrated by simulating the evacuation of an airport, clearly identifying specific considerations that other models do not take into account, and making comparisons with those other models. The key concept in [4] provided us with the demonstration and record of how to take specific ideas (panicked crowd, random movement, etc.) in specific situations (evacuation with children, authorities, as well as emotional and social aspects) and develop a model and simulation. Ideas such as Spread of Knowledge (SoK), Emotional Contagion, and Social Comparison Theory (SCT) were borrowed from this paper. These three concepts were broken down into specific behaviors, and these behaviors were mapped to a finite state machine as part of a DEVS model. The work presented in [5] provides an evaluation of existing techniques for modelling the egress of crowds in emergency situations, looking at particle based models in particular. It categorizes the existing literature on crowd behavior into five distinct categories, and then evaluates various modelling techniques on their consideration of these categories. In the context of this paper, the suite of models produced were made in cognizance of these five categories, understanding the various components and elements that drive crowd behavior during emergencies. The Review of Building Evacuation Models technical report [6], produced by the National Institute of Standards and Technology, provides a comprehensive review of thirty existing building evacuation models so that building designers can make informed choices when selecting an evacuation model for their design. This technical note demonstrates that many of the evaluated models tend to focus on either behavioral simulation, or movement simulation; this specialization allows the models to be applied for very specific usages.

Cell-DEVS extends the concept of the Cellular Automata formalism to build large-scale multi-dimensional models of defined neighbourhoods where each cell in the neighbourhood is represented by a discrete value or DEVS models. Each individual DEVS cell in the neighbourhood is evaluated on a periodic basis using a set of defined rules, and these rules govern the state of each cell, and how each cell influences the other cells in its local neighbourhood. Cell-DEVS is driven by a mathematical formalism that defines the components of a Cell-DEVS model, and how these components interact in terms of logic and operations. The egress models presented in this paper compromised of two dimensional neighbourhoods of cells that define the local area to be evacuated. Each neighbourhood represents a single plane, with each plane providing different information about the local area being evacuated. These planes overlay each other to create a multi-dimension Cell-DEVS model, with more complex models requiring more dimensions to capture the required data. Each plane in each model has its own set of rules governing the behavior of all cells in the local area.

Our Cell-DEVS models were constructed, compiled, and simulated using the CD++ toolkit, an eclipse plugin that was built by Wainer and the DEVS community at Carleton University that both utilizes and compliments the programming capabilities of the C++ programming language[3]. While the DEVS models themselves rely on the standard C++ constructs with .cpp source files and .h include files, the Cell-DEVS models make use of constructs created specifically for the toolkit, with its own specific syntax that needs to be adhered to. Segments demonstrate how to construct a CD++ source file that is representative of a Cell-DEVS model will be illustrated in the next section.

III. Emergency Evacuation Modeling

There is a large quantity of literature available exploring the mechanics behind egress and evacuation during emergency situation. From the psychology of human behavior, to the predictability of human movement, to the determination of bottle-necks and slow-moving areas within a building, this literature has been built upon by multiple researchers over the course of multiple iterations so that better sets of controls and procedures can be developed to aid in the overall construction of buildings and the development of evacuation plans.

There are a number of commercially available egress and evacuation models designed to aid in building construction and evacuation planning, and there is literature out there that has reviewed many of these products to determine the applicability of each model. It begs the obvious question, why do we need to produce a set of Cell-DEVS models for egress and evacuation? One of the common threads picked up in the model reviews by Kuligowski and Peacock [6] is that many of the commercial models are designed to cater for specific situations, with little room for customization. There are few models that offer a more generic approach with the goal of wide application across a variety of situations. What this research effort aims to do is to provide all the building blocks necessary so that models can be constructed for a variety of situations for a variety of specific purposes. Easy and rapid customization and extension is the key, but first we need to know what the basic building blocks are, demonstrate that they can work individually, demonstrate that they can work when integrated, and demonstrate that they can be integrated in a specific fashion for specific purposes.

A. Movement Modeling

The first step towards building evacuation models is understanding the different patterns that define human movement. Some occupants will know where the exits are located because of their familiarity with the area, and will move towards the nearest exit using the shortest path available. Some occupants will have no idea where the exits are because they have never been in the building before, and will move around erratically or randomly. Authority figures may be following predefined patrol paths, search for any occupants who have not yet evacuated. Some occupants who are unfamiliar with the area will look for other occupants and begin to follow their lead. Some occupants will rely on directional aids to define their movements such as exit signs and evacuation maps if available.

B. Behavior Modeling

Once we understand the different patterns of movement that are exhibited by occupants in an evacuation, we can start to look at how different occupant behaviors will influence these behaviors. If the path to the exit is blocked by too many people,

occupants may experience heightened anxiety leading to a panic state, where movement will change from controlled to erratic. An occupant in a panic state may then influence other occupants in the immediate surrounds, who may also shift into a panic state. A person who does not know where they are going may have their movement pattern changed after interaction with an authority figures. who is patrolling the area to be evacuated.

C. Categorization

There are a number of common elements for the evacuation and egress models that need to be defined. The points below show the categorizations that are used in this work. These categories have been constructed based on the efforts in [4][5][6].

Random Movement: The occupant selects a random direction and moves in that direction. Not often directly applicable to real life scenarios but does influence other behaviors such as panic. This movement is reserved for the rare cases where occupants have no knowledge of where the exit is, and are not able to deduce where it might be based on the environment.

Learned Exit Knowledge: The occupant knows where the exits are and can determine the shortest route to the nearest exit. Direction and movement will be based on this knowledge. Often applies to employees at workplaces who are regularly drilled on evacuation procedures.

Observed Exit Knowledge: The occupant can determine where the exits are by observing doors and exit signage, or following evacuation maps. Direction will require observations of these signs, while movement will continue in the indicated direction until new directional information is observed. In a public area such as an airport, this will initially be the most likely movement type.

Directed Movement: Authority figures in key positions can provide information to occupants who do not have knowledge of the exits. They can also control the flow of occupants to desired exits, using their knowledge of events such as whether or not an exit is blocked. Authority figures can also reduce panic within an environment. Finally, authority figures can also be deployed along fixed routes to determine that all occupants have evacuated the building.

Follow-the-herd Movement: This is a psychological behavior that can override an occupant's knowledge of the exit. If an occupant observes a crowd of people moving in a specific direction, they may abandon their knowledge of the exits and decide to follow what the crowd is doing. If an occupant doesn't know where an exit is, they will almost certainly follow the crowd. This type of behavior can often be the result of counter-flow, where people are made unsure due to the movement of other people opposite to the direction of the exit.

Panicked Movement: Another psychological behavior, panic can result in a rush of adrenaline, causing people to behave and move erratically at a faster pace. We have used random behavior and twice the movement to model the panic behavior.

IV. Cellular Discrete Event Approach

The selected modeling and simulation methodology for this work is the Cellular Discrete-Event System Specification (Cell-DEVS) formalism [2]. A model defined by Cell-DEVS is a cellular grid where each cell represents an independent agent. Agents communicate and interact with each other throughout the simulation by sending/receiving messages. The behavior of each agent is expressed using a state-machine. Agents can be stationary or mobile meaning they can change their position within the grid. Each cell defines a surrounding neighborhood that affects its state value. Whenever a computation is performed and the cell's value is modified, all its neighboring cells get an update to re-evaluate their states accordingly. A cell's value changes as a result of a simple local computation based on the current state of the cell and its immediate neighbors as dictated by the Cell-DEVS specification. Fig. 1 illustrates a typical Cell-DEVS grid and the neighboring definition. Each cell can choose to have as many direct or indirect neighbors.

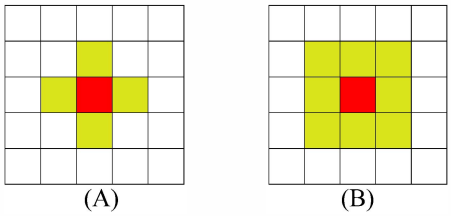

Fig. 1. Cell Neighborhoods: (A) Von Neumann (each cell takes up to four neighbors from North, South, East, or West), (B) Moore (each cell takes up to eight neighbors from North, South, East, West, Northeast, Northwest, Southeast, or Southwest).

A Cell-DEVS model is defined by:

$$TDC = < X, Y, I, S, \theta, N, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >,$$

where X is a set of external input events; Y is a set of external output events; I represents the model's modular interface; S is the set of sequential states for the cell; θ is the cell state definition; N is the set of states for the input events; d is the delay for the cell; δ_{int} is the internal transition function; δ_{ext} is the external transition function; τ is the local computation function; λ is the output function; and D is the state's duration function. The modular interface (I) represents the input/output ports of the cell and their connection to the neighbor cell. Communications among cells are performed through these ports. The values inserted through input ports are used to compute the future state of the cell by evaluating the local computation function τ. Once τ is computed, if the result is different from the current cell's state, this new state value must be sent out to all neighboring cells informing the state change. Otherwise, the cell remains in its current state and therefore no output will be propagated to other cells. This will happen when the time given by the delay function expires. Finally, the internal, external transition functions and output functions (λ) define this behavior. 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.

CD++ [3] is an open-source object-oriented modeling and simulation environment that implements Cell-DEVS

theories in C++. The tool provides a specification language that defines the model's coupling, the initial values, the external events, and the local transition rules for Cell-DEVS models. CD++ also includes an interpreter for Cell-DEVS models. 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 the border. The cell's local computing function is defined using a set of rules with the form *postcondition delay {precondition}*. These indicate that when the *precondition* is met, the state of the cell changes to the designated *postcondition* after the duration specified by *delay*. If the precondition is not met, then the next rule is evaluated until a rule is satisfied or there are no more rules. CD++ also provides a visualization tool, called *CD++ Modeler*, which takes the result of the Cell-DEVS simulation as input and generates a 2-D representation of the cell space evolution over the simulation time. This feature of the tool provides an interactive environment allowing for visual tracking of the mode's evolution.

V. Proof of Concept and Analysis of Basic Models

The purpose of this section is to present a variety of small and simple evacuation models that will be built upon, expanded upon, and integrated incrementally. Five basic evacuation models will be presented. Integration of these models and introduction of complications within each model are left for future research effort.

A. Models Properties and Core Concepts

These evacuation models are designed to represent simple movements utilizing a single rule set with no behavioural driven changes of state. These models will be simulated using a small four-room building with a limited number of occupants as demonstrated in Fig. 2.

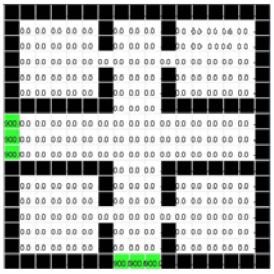

Fig. 2: Basic simulation floor map.

In this floor map, the 0 value states are colored white and represent available space, the -1 value states are colored black and represent walls or obstacles, and the 900 value states represent the exit. Occupants will be represented by states ranging from 1 to 90, depending on their movement type and the current direction they are moving in. Table 1 summarizes these cell values.

Table 1: Basic Model Legend

State	Meaning	Color
-1	Wall/ Obstacle	Black
0	Free Space	No Color
1 - 100	Occupant	Blue
900	Exit	Green

All of the models are defined by their rule sets, with collections of rules designed to implement specific mechanics. Given the large quantity of rules governing each model, and for the purpose of brevity, direct relationship between rule set and mechanic will only be explored in detail for the Core Concepts and the Random Movement model. For the rest of the models, the specific mechanics and the arrival at a rule set solution will be discussed, but much of the detail will remain embedded within the comments in the model source code (*available open-source per request)*.

The development of Cell-DEVS models and the rules that govern them, in the context of this paper, are driven by the following core concepts:

- Cells come in four generic types – obstacles, empty space, exits, and occupants. Obstacles include walls, desks, or anything that an occupant needs to navigate around.
- Occupant cell states can be used to determine the movement type of an occupant and the direction they are travelling. State values for occupants can range from 1 – 800, with directional data embedded in the 1's and 10's, and movement type data embedded in the 100's.
- Directional data is based on eight compass points, with directions equating to the following values: 1 – North, 2 – North East, 3 – East, 4 – South East, 5 – South, 6 – South West, 7 – West, 8 – North West. Fig. 3 provides a visual representation of this directional data.

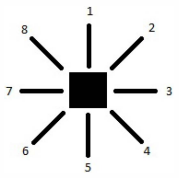

Fig. 3: Compass detailing directional data.

- When a cell has been evaluated and an applicable rule has determined a change in direction, the state of the cell will be the desired direction multiplied by ten. After a move in the chosen direction has been successfully completed, the state of the cell will be divided by ten. The reason for this is because of an anomaly within the priorities of the rule set. Rules are evaluated in priority order, so if the state was never multiplied by ten and direction rules were written first, each periodic evaluation of the cell state would result in a change of direction, and the rules governing movement in the chosen direction would never be reached. Single digit numbers indicate to the rule set that the cell is ready to evaluate its direction, while multiples of ten indicate that the cell is ready to move in the desired direction.

- Given the period evaluation of each cell in the neighbourhood, collision detection becomes a problem when the rule set drives two occupants to enter the same cell at the same time. This problem can be overcome by manipulating the neighbourhood, allowing a cell to predict the movement of other cells more than two spaces away. When two cells desire to occupy the same space, the cell that occupies that space first is determined based on priority. The priority order numerically maps to the directional data, with one being the highest priority direction through to eight being the lowest priority direction.
- The preferred neighborhood shape for evaluating each cell on the zero-plane will be a 5x5 grid with the cell to be analysed residing at the center. For cells residing on other planes, the neighborhoods defining how they are to be analysed, and how they will interact with other planes, will be discussed in the applicable model.
- Each rule set requires a single rule that, when evaluated, does nothing except maintain the current state of the cell. This is for the case where an evaluated cell does not meet the criteria for any of the other listed rules in the rule set.

B. Random Movement Model

In this evacuation model, the occupants of the building will choose a random direction and step forward provided there are no obstacles in the way. This very simple mechanic will play a part in future models and simulations that involve panic states. Four sample simulation scenarios with floor maps and occupants are illustrated in Fig. 4.

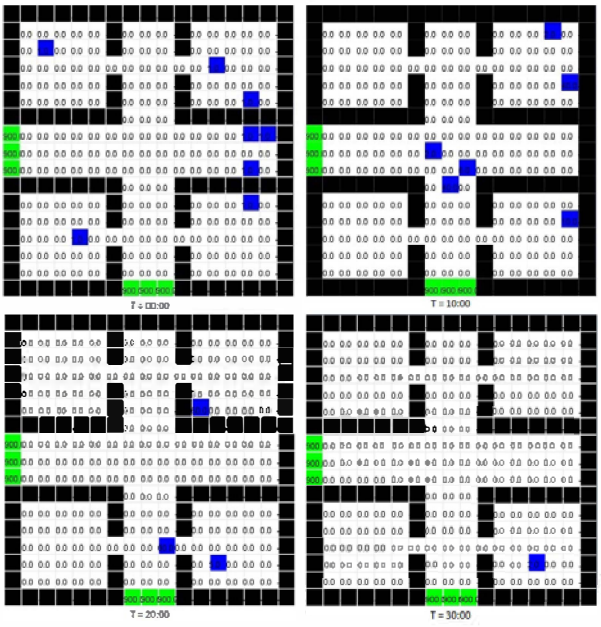

Fig. 4. Random movement model.

In terms of constructing a rule-set that defines this model, we need to only consider cellular interaction on a single plane, but we need to explore five separate movement mechanics:

- **Exit Area** – this mechanic evaluates whether an occupant cell is next to a marked exit, and if it is, it removes the occupant from the defined area. This mechanic is listed first because CD++ evaluates rules top to bottom, and this is a one off mechanic that may never be utilized if it were to be placed below a regularly used mechanic like Get Direction.
- **Get Direction** – this mechanic evaluates whether an occupant cell is ready to obtain a direction, then uses the randInt function to randomly choose an integer between one and eight. The value of the integer represents the direction of movement as defined by Fig. 2. The rule that defines this movement is as follows:
- **Enter Cell** – this mechanic evaluates whether or not a cell is empty, and if it is empty, it determines if there is a neighboring occupant cell ready to move into that empty cell. This mechanic also determines if there are multiple cells desiring to enter that empty cell, and will only let the highest priority cell enter
- **Leave Cell** – this mechanic evaluates whether or not an occupant cell is ready to move, and if it is ready to move, it determines if the neighboring cell in the desired direction of movement is empty. This mechanic uses the 5x5 neighborhood to determine if there are any other cells looking to enter the desired empty cell, and if there are, the mechanic evaluates the priority of those cells against the current cell to determine if the current cell is able to move into the desired empty cell.
- **Try Again** – this mechanic is enacted if an occupant cell cannot move in its chosen direction because the cell it wants to move to is already occupied. The mechanic resets the cell state to one, allowing the Get Direction mechanic to set a new random direction.

C. Optimal Movement Model

In this evacuation model, the occupants of the building know where the exits are and follow the shortest path to the nearest exit. In this model and simulation, the occupant will determine their direction of movement by accessing how far away they are from nearest exit based on their knowledge of the area.

In terms of constructing a rule-set that defines this model, we need to consider cellular interaction on two planes, with the zero plane providing the floor-map and positional data of the occupants, and the first plane providing data on the distance of every cell in the neighborhood relative to the nearest exit. The local neighborhood used to evaluate cells on the first plane is a 3x3 grid. The Enter Cell, Leave Cell, and Exit Area mechanics are maintained from the Random Movement Model, while the Get Direction mechanic is modified and a new mechanic is introduced:

- **Get Direction** – this mechanic evaluates whether an occupant cell is ready to obtain a direction, and then accesses the corresponding cells on the first plane to determine the direction if it is ready. The corresponding cell on the first plane states how many cells there from the

current position to the nearest exit, and the mechanic scans the first plane neighborhood looking for the next cell that is one step closer to the exit. The mechanic then uses this knowledge to determine the direction the occupant cell needs to move.

- **Maintain Current Direction** – this mechanic evaluates whether an occupant cell is ready to obtain a direction, and then accesses the corresponding cells on the first plane. The mechanic only checks the adjacent first plane cell in the currently held direction, and if that cell represents a step closer to the exit, the current state of the zero plane occupant cell is maintained.

D. Directional Movement Model

In this evacuation model, the occupants of the building are able to find the exit points by following directional cues such as signage. In this model and simulation, the occupant will determine their direction of movement by assessing the information at hand, and continue to move in that direction until the information at hand indicates a change of direction is required. On this small scale basic model it is assumed that all exit signage is visible, however, for larger scale models the visibility of the signage may be lost in certain areas, which would require the occupant to select their own direction and begin their search for directional information.

In terms of constructing a rule-set that defines this model, we need to consider cellular interaction on two planes, with the zero plane providing the floor-map and positional data of the occupants, and the first plane providing data on the direction to travel. The local neighborhood used to evaluate cells on the first plane is a 3x3 grid. All five mechanics are maintained from the Optimal Movement model, with the Get Direction and Maintain Current Direction mechanics slightly modified:

- **Get Direction** – this mechanic evaluates whether an occupant cell is ready to obtain a direction, and then accesses the corresponding cells on the first plane. The first plane provides directional data as defined by Fig. 3, with direction of the occupant cell set to whatever value is being held by its corresponding first plane cell.
- **Maintain Direction** – given that this model looks to represent visible signage, it stands to reason that not all cells on the first plane will provide directional data. This mechanic evaluates whether or not directional data has been provided by the first plane cell corresponding with a zero plane occupant cell, and if there is no directional data provided, the direction of the occupant cell is maintained.

E. Patrol Movement Model

In this evacuation model, a person of authority will follow an establish path, patrolling the building to determine whether or not everyone has been evacuated. In this model there are two planes, a plane that contains the floor map and the occupant positions, and the plane that contains the route for the person of authority to follow during their patrol. For this basic model the interactions with other occupants will not be handled, this is just a proof of concept to show that occupants can follow a patrol path. There is a variation on method where people of authority are placed in key areas and thoroughfares, providing guidance towards the nearest or best exit, and providing a calming influence for impatient or panicked occupants.

In terms of constructing a rule-set that defines this model, we need to consider cellular interaction on two planes, with the zero plane providing the floor-map and positional data of the patrolling occupant, and the first plane providing a patrol route for the patrolling occupant to follow. The local neighborhood used to evaluate cells on the first plane is a 3x3 grid. The Enter Cell, Leave Cell, and Exit Area mechanics are maintained, while the Get Direction mechanic is modified:

- **Get Direction** - this mechanic evaluates whether an occupant cell is ready to obtain a direction, and then accesses the corresponding cells on the first plane to determine the direction if it is ready. The corresponding cell on the first plane states how many cells the patrolling occupant has progressed on their patrol route, and the mechanic scans the first plane neighborhood looking for the next cell on the patrol route. The mechanic then uses this knowledge to determine the direction the occupant cell needs to move.

It should be noted that in this specific implementation of the Patrol Movement Model, there is an issue if the patrol path crosses over itself. This issue can be rectified through the creation of specific rules that update the patrol path each time a cross over is navigated. For the floor map provided by Fig. 2, there were twelve cells identified on the patrol route that would be passed over twice by the patrolling occupant, and these cells have their own corresponding rule to handle the crossover.

F. Follow the Herd Movement Model

In this evacuation model, the occupants of the building who observe the formation of a herd of occupants will choose to follow in the same direction as the herd. This is a psychological phenomenon, where an occupant makes an assumption that a group of people moving with purpose must know where they are going. In this model and simulation, an occupant will observe a group of people and start moving in the same direction as those people. This model and simulation will also take into account the actions of a leader who will determine the overall direction for the herd to travel. For this model, the leader will only change direction when confronted with a wall, but future iterations of this model should consider more sophisticated direction finding techniques for the leader to apply. In terms of constructing a rule-set that defines this model, we only need to consider cellular interaction on a single plane, with the zero plane providing the floor-map and positional data of the occupants, however, the model will be setup using multiple planes as its expected that this model will form the basis for future iterations that will make use of multiple planes. A major difference for this model is that there are three different types of occupants that will be represented by this model – stationary occupants, leading occupants, and following occupants. In order to differentiate between these types of occupants, the state value for occupants will make use of the hundreds column:

- 001 - Stationary Occupant – Red

- 101-180 – Following Occupant – Yellow
- 201-280 – Leading Occupant – Blue

This model makes use of thirteen mechanics to drive the evaluation of cells. Six of these mechanics are the Enter Cell, Leave Cell, and Exit Area mechanics from the previous models, having been adapted for leading occupants and following occupants. The mechanics detailed below are new or modified for this model:

- **Leader: Get Direction** – this mechanic evaluates whether a leading occupant needs to change direction because of an approaching wall. For this mechanic, the design decision was made for the leading occupant to change direction in an anticlockwise manner until there is no observable wall blocking the path forward.
- **Leader: Keep Current Direction** – this mechanic evaluates whether a leading occupant can maintaining its current direction of movement
- **Transition from Stationary to Following Occupant** – this mechanic evaluates the neighborhood surrounding a stationary occupant, and if a leading or following occupant is observed within this neighborhood, the stationary occupant will become a following occupant.
- **Follower: Get Direction from Leader** – this mechanic evaluates the neighborhood surrounding a following occupant, looking to observe the direction of movement being used by a leading occupant. The following occupant will adjust its direction of movement relative to the leading occupant, moving out of the way if it is blocking path of a leading occupant, or matching the same direction if it is beside or behind the leading occupant.
- **Follower: Get Direction from Another Follower** – this mechanic evaluates the neighborhood surrounding a following occupant, looking to observe a situation where there are only following occupants and no leading occupants in the neighborhood. The following occupant will set its direction to the same as any other following occupants in its neighborhood, with direction priority the same as collision detection priority if there are two following occupants with different directions in the neighborhood.
- **Follower: Keep Current Direction** – this mechanic evaluates the neighborhood surrounding a following occupant only when there are no leading or following occupants in the neighborhood. If there is no wall blocking progress, the following occupant will continue to move in the same direction.
- **Follower: Change Direction** – this mechanic evaluates the neighborhood surrounding a following occupant only when there are no leading or following occupants in the neighborhood, and there is a wall blocking progress. The direction of movement will change in an anticlockwise direction until there is no observable wall blocking the path forward.

VI. Results and Analysis

The results and analysis of the simulations performed based on the models constructed are presented in this section. The analysis explore the size of the model in lines of code (LOCs – including comments and blank lines), quantity of Cell-DEVS rules, the simulation runs, and the behavior exhibited by the models through the in-built CD++ visualizer. It is important to note that during the experimentation phase each model and simulation was executed five times. The reason for multiple runs is to evaluate the performance of the CD++ toolkit, and to demonstrate the integrity of the seed that the CD++ toolkit has been built around. Each simulation run was replicated five times, the timing results shown here represent the averaged values. Each simulation was run until all occupants evacuated the area. This was easily observable by looking at the visualizations and following the crowd movement until no occupant cell was left on the simulated floor area. The overall egress time for every model was captured by CD++ Modeler (the visualization interface indicated the total simulation time). The log files would also report these timings. These simulations runs are available on our youtube channel [7].

Table 2 provides statistical results of the five explained models over five different metrics. Analysis of these data are given below each graph in the following paragraphs.

TABLE 2. EGRESS MODELS STATISTICS.

Model	LOC	Rules	Simulation Time (Seconds)	Average Computation Time (seconds)	Average Memory Usage (MB)
Random Movement	26	28	2155	18.54	23.67
Optimal Movement	95	35	27	1.77	7
Directional Movement	93	35	34	2.51	10
Patrol Movement	97	39	178	2.05	8
Follow the Herd	207	116	76	3.88	12.33

Fig. 6 data demonstrates the iterative development approach taken when building the models. Each of the later models was an iteration of the previous model, adding new pieces of functionality as required.What these values do not show is the complexity of each rule, and as the iterations continued, the complexity required for each rule would start to increase. Complexity seemed to equate to character length, so an option might be to average the number of characters making up all the rules. Looking at the raw data for simulation versus computation times in Fig. 7, we can see a correlation between total length of the simulation and the average time taken to compute each model. It seems to hold true for most of the models, but it is not supported by the Random Movement model statistics, which has the largest simulation time of all models. The missing connection here comes from the rule sets in the models, and the number of cells in the model, which both have a proportional contribution to computation time. As more and more rules are introduced, and multiple planes are introduced, the amount of time required to process every rule for every step increases.

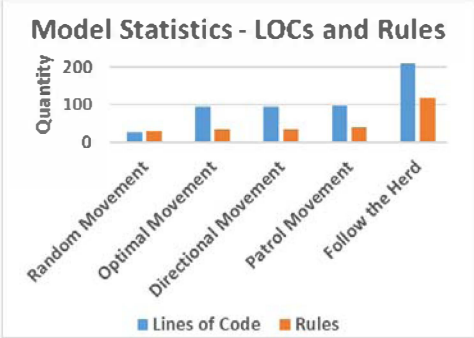

Fig. 5. Model statistics - LOCs and rules.

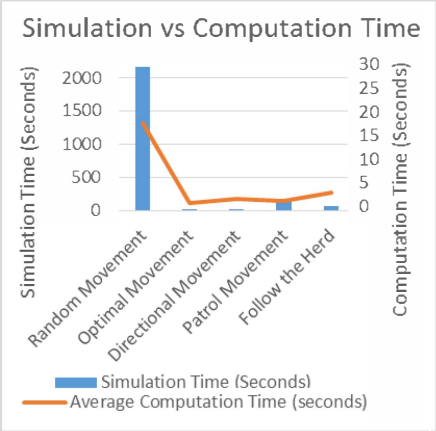

Fig. 6. Simulation vs computation time.

While the memory usage was recorded for each simulation run for each model, the results are inconsistent and are not reliably indicative of any trend. Looking at the plot at Fig. 8, a proportional relationship can be observed between Computation Time and Memory Usage. These values were all collected in a single CD++ session; however, the size of proportionality is dependent on the specifications of the computer hardware running the simulations, and the length of the CD++ session. Measuring memory requirements as a function of session length may be a candidate for future work, especially if memory issues are found in the CD++ toolkit in the future.

VII. Conclusion

This paper provides the ongoing Cell-DEVS research community with several expanded and unique designs for occupant evacuation scenarios in an Egress Modeling Suite with conclusive detail and accurate findings. Model specifications and brief cell-DEVS rules were illustrated. Simulation results were conducted to demonstrate effectiveness of evacuation plans to guide first responders and decision makers in evacuating crowd under emergencies. Egress modeling is a complex phenomenon that requires behavioral and psychological considerations. The work presented here is the first step in building a general-purpose and customizable modeling and simulation database for emergency evacuations.

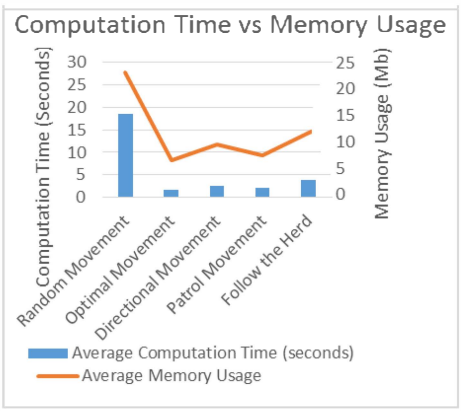

Fig. 7. Computation time vs memory usage.

Evacuation complications such as panic, lost crowd, related group of people (members of a family), disabled people, seniors, and children are more sophisticated factors that are currently being considered by the authors. Initial data have been collected by applying the models presented here to a realistic situation in an airport layout. Future publications of the authors will include details of their new research data. Simulation videos and models' source code can be found at [7 and 8]. Related Cell-DEVS research can be found at [9].

REFERENCES

[1] Pan, X. Computational Modeling of Human and Social Behaviors for Emergency Egress Analysis. Stanford University, Standford, 2006.

[2] Wainer, G. 2009. Discrete-event modeling and simulation; a practitioner's approach. Taylor & Francis.

[3] Wainer, G. 2002. CD++: A Toolkit to Develop DEVS Models. Software – Practice and Experience, 32(13), pp. 1261- 1306.

[4] Tsai, J., and et.al. ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions and Social Comparison. *Proc.* International Foundation for Autonomous Agents and Multiagent Systems, The 10th International Conference on Autonomous Agents and Multiagent Systems (2011).

[5] Radianti, R, and e. al. Crowd Models for Emergency Evacuation: A Review Targeting Human-Centered Sensing. *Proc.* System Sciences (HICSS), 46th Hawaii International Conference on System Sciences (2013).

[6] Kuligowski, E. D., and Peacock, R. D. Technical Note 1471: A Review of Building Evacuation Models. National Institute of Standards and Technology, Washington (2005).

[7] Egress Simulation Videos. Available at: https://www.youtube.com/channel/UC3TjIsu41ahPqDiVAQBlmmA/videos [last access: February 2015]

[8] Egress Cell-DEVS Models Source Code. Available at: https://github.com/sjafer/Egress_Cell-DEVS_Models.git [last access: February 2015].

[9] Cell-DEVS Publications form ARSLab. Available at: http://tinyurl.com/m569a8q [last access: February 2015].