

# Emergency crowd evacuation modeling and simulation framework with cellular discrete event systems

Shafagh Jafer and Ryan Lawler

## Abstract

First responders and security personnel face many challenges to safely evacuate crowded environments. Conducting frequent physical and table-top exercises are neither feasible nor economical. This is where modeling and simulation comes into play in providing a risk-free and economical method to practice various evacuation strategies, train first responders, and provide accurate decision-support and emergency guidance. With the aid of formal methods, here we present a suite of various egress strategies build on top of an open-source modeling and simulation environment. We use Cellular Discrete-Event System Specification (Cell-DEVS) formalism to explore emergency evacuation scenarios by building 12 egress models representing aspects of human behaviors under emergencies and the activities of authorities in guiding the crowd. The proposed framework explores random and controlled human movement, as well as implementing psychological conditions such as herd following and panicked states. We provide results analysis to compare evacuation speed under various egress methods, allowing for decision making when using the suite for training purposes. The outcome of this work is available on a public repository to serve the DEVS community, researchers, and public safety authorities interested in emergency evacuation simulations.

## Keywords

Crowd evacuation, egress simulation, emergency modeling, Cellular Discrete-Event System Specification

## 1. Introduction

Efficient crowd guidance methods are one of the key concerns of safety personnel under emergency evacuations. Handling such events is only possible through precise estimation and training. To mitigate the enormous financial and human loss caused by emergencies, modeling, simulation, and visualization technologies come in handy. Such technologies should address multiple interdependent aspects of evacuation scenarios by integrating different aspects of an emergency situation. Aiming at complementing and to some extent replacing physical and table-top emergency exercises, such frameworks will need to also provide real-time modeling and analysis capabilities for the emergency risk analyst. While modeling and simulation of human behavior during emergency situations is not a new concept, existing efforts have already helped to improve safety and aid in the development of evacuation methodologies. However, this application of modeling and simulation is far from exhaustive and there exists much room for improvement. 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.<sup>1</sup> A flexible modeling and simulation construct would be effective in this situation. This is where the Cellular Discrete-Event System Specification (Cell-DEVS)<sup>2,3</sup> modeling formalism comes into play. The Cell-DEVS formalism has been successfully used to model systems whose behavioral state changes based on trigger events. Here we explore the extension of DEVS beyond the modeling of event-driven systems but focusing on the application of DEVS to changes in human behavior as driven by emergency events. This paper will present our efforts in developing a suite of various crowd emergency behaviors in a variety of situations. We report on the development of 12 egress models and simulations that can aid in the

---

Department of Electrical, Computer, Software, and Systems Engineering,  
Embry-Riddle Aeronautical University, USA

### Corresponding author:

Shafagh Jafer, Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, 600 S. Clyde Morris Blvd., Daytona Beach, FL 32114, USA.  
Email: jafers@erau.edu

development of crowd guidance optimization methods in emergency situations. The models generated in this work are open-source and available to the DEVS community at Egress Cell-DEVS Models Source Code.<sup>4</sup> While our initial efforts have been previously published,<sup>5</sup> here we provide the details of additional egress models.

The paper is organized as follows: Section 2 will provide an overview of crowd modeling and provide a brief literature survey over the topic. Section 3 introduces the discrete-event approach adopted by this work. Section 4 discusses common egress patterns. The various egress models and their Cell-DEVS details are presented in the Section 5. Section 6 discusses the results of various simulation runs over the developed egress models. Finally, Section 7 summarizes the work and provides some future insights.

## 2. 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<sup>6</sup> provides an evacuation simulation tool designed to capture the intricacies when trying to evacuate families from unfamiliar public spaces. In Tsai et al.,<sup>6</sup> 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 other models. The key concept in Tsai et al.<sup>6</sup> 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. 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 by Radianti et al.<sup>7</sup> provides an evaluation of existing techniques for modeling the egress of crowds in emergency situations. It categorizes the existing literature on crowd behavior into five distinct categories, and then evaluates various modeling techniques on their consideration of these categories. The Review of Building Evacuation Models technical report,<sup>8</sup> produced by the National Institute of Standards and Technology, provides a comprehensive review of 30 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. There exist a significant number of literatures on egress modeling and simulation. Given the nature of Cell-DEVS and its benefits over the traditional Cellular Automata (CA),<sup>9</sup> in this work we focused on CA-based research to derive some of our models' assumptions and terminology.<sup>10-13</sup> For details on different research, we suggest readers explore the work presented by Peacock and Averill.<sup>14</sup> Besides, given the cellular space approached selected for our work, collision avoidance (two agents trying to occupy one vacant cell) has been addressed by assigning directional priority. The work presented by Song et al.<sup>15</sup> takes a different approach in resolving conflicting cells by taking into account social and psychological factors.

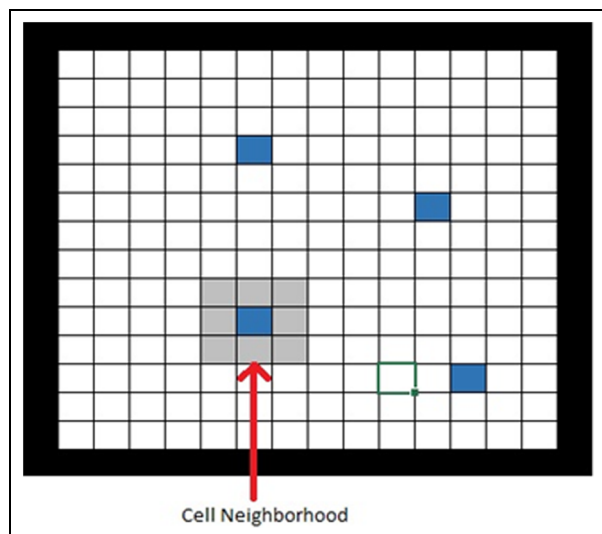
We have chosen Cellular Discrete Event System Specification (Cell-DEVS)<sup>3</sup> to implement our egress models. Cell-DEVS is an application of DEVS<sup>2</sup> that establishes a grid of cells where each cell is in effect an individual DEVS model. The Cell-DEVS formalism is descendent from the CA<sup>9</sup> formalism, requiring designers to construct a model that includes a neighborhood of cells that each take on a finite value and a set of rules that govern changes to these cells. For each rule defined by the model, every cell in the neighborhood is evaluated only if it has taken a new value compared to the last evaluation phase. This improves the computation performance significantly, as only a subset of the grid will require re-evaluation as opposed to CA theory, where all cells are re-evaluated at every evaluation phase. The combination of these characteristics makes Cell-DEVS a great candidate for modeling egress scenarios where a large number of interactions occur among a large number of agents. On the other hand, the grid-based approach of the Cell-DEVS formalism can be used to represent the physical bounds or a simulated location, such as the physical components of a building and the people inside, while the rule-based approach can be used to represent behaviors and interactions between individual cells, such as people reacting to a blocked exit during an emergency.

## 3. Methodology

This work explores specific usages of the Cell-DEVS formalism that can be utilized to produce models and simulations for specific methods of egress and evacuation. This section will first provide an overview of Cell-DEVS and how it is intended to be utilized for egress modeling purposes. Then, we present an overview of our egress methods.

### 3.1. Cell-DEVS formalism

Cell-DEVS extends the concept of the CA formalism to build large scale multi-dimensional models of defined



**Figure 1.** Two-dimensional cellular neighborhood.

neighborhoods where each cell in the neighborhood is represented by a discrete value. Each individual DEVS cell in the neighborhood is evaluated on a periodic basis (defined by a delay parameter) using a set of defined rules. The rules govern the state of each cell and how each cell influences the other cells in its local neighborhood. Figure 1 demonstrates how a two-dimensional cellular neighborhood can be represented.

As mentioned above, 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. For a basic DEVS model, the formalism is expressed as follows:

$$M = \langle X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle \quad (1)$$

- $X$  = a set of input events driving the model.
- $Y$  = a set of output events driven by the model.
- $S$  = a set of states that the model can possess.
- $\delta_{\text{int}}$  = a set of internal events that can trigger specific changes of state.
- $\delta_{\text{ext}}$  = a set of external events that can trigger specific changes of state.
- $\lambda$  = the output function for the model driven by the current state of the model and triggered by a state transition.
- $ta$  = the time advance function. When internal and external events have not occurred, the evaluation of a model, leading to a potential change of state, is triggered after a specified amount of time has elapsed.

A Cell-DEVS model is considered to be a network of coupled DEVS models, so the formalism defining a Cell-DEVS model is slightly modified from the basic DEVS

model formalism. For a Cell-DEVS model, the formalism is expressed as follows:

$$TDC = \langle X, Y, S, N, type, d, \tau, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, D \rangle \quad (2)$$

where  $X$ ,  $Y$ ,  $S$ ,  $\delta_{\text{int}}$ ,  $\delta_{\text{ext}}$ , and  $\lambda$  represent the same definitions as the basic DEVS model.

- $N$  = a set of input values for each cell, either provided by external events or neighboring cells.
- $type$  = the type of delay, defining how cells will experience state transitions.
- $d$  = the length of time for the delay, driving a cell's state transition. The  $type$  and  $d$  values work together in a similar fashion to the  $ta$  value from the basic DEVS model.
- $\tau$  = the local computing function for an individual cell that takes the set of input values  $N$  and evaluates them in order to determine what the next state of that cell will be.
- $D$  = the set of cells that comprise the Cell-DEVS model.

An important part of defining each egress and evacuation model is the definition of the total model neighborhood,  $D$ . The work presented here will define models comprised of two-dimensional neighborhoods of cells that define the local area to be evacuated. Each neighborhood represents a single plane, with each plane providing different information about the local area being evacuated. These planes will 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. The rule specifications are covered in Section 5.

### 3.2. CD++ toolkit

For this work Cell-DEVS models were constructed, compiled, and simulated using the CD++ toolkit, an open-source Eclipse plugin that both utilizes and complements the programming capabilities of the C++ programming language.<sup>16</sup> 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 their own specific syntax that needs to be adhered to. The following sections demonstrate how to construct a CD++ source file that is representative of a Cell-DEVS model.

### 3.3. Cell-DEVS model development

The beginning of a CD++ Cell-DEVS model source file provides the initial model construction information that the

```

[top]
components : Rand

[Rand]
type : cell
dim : (17, 17)
delay : transport
defaultDelayTime : 1000
border : nowrapped

```

**Figure 2.** Cellular Discrete-Event System Specification header (model file).

CD++ toolkit will use when compiling the model and executing a simulation of the model. Figure 2 shows the initial declarations for the Random Movement model constructed as part of this work.

The first required step is to establish the [top] section, which requires the developer to define all of the neighborhoods or individual DEVS models that make up the overall model. The next step is to start defining the individual components established in the [top] section, setting up the neighborhood parameters that will be used by the model as follows.

- The **type** parameter refers to the type of DEVS model being developed, which are Cell for the suite of models developed by this research project. This is not to be confused with the type parameter from the Cell-DEVS formalism.
- The **dim** parameter refers to the dimensions of the neighborhood to be modeled, and this will be specific to area being modeled, and the type of model being used. This parameter establishes the bounds for the  $D$  parameter from the Cell-DEVS formalism.
- The **delay** parameter in CD++ refers to two types of state transition defined by the type parameter in Cell-DEVS formalism. A “transport” delay is a state transition resulting from an input command after a specified period of time, while an “inertial” delay is a state transition resulting from an input command only if the input command remains the same for a specified period of time.
- The **defaultDelayTime** parameter establishes periodic time steps, governing the frequency at which the model is evaluated. The units for this parameter are milliseconds. All of our models reported here have set the *defaultDelayTime* to 1000 milliseconds. This parameter establishes the default value for the parameter  $d$  from the Cell-DEVS formalism,

```

Rand.ma X
%3x3 Neighborhood
neighbors : Rand(-1,-1) Rand(-1,0) Rand(-1,1)
neighbors : Rand(0,-1) Rand(0,0) Rand(0,1)
neighbors : Rand(1,-1) Rand(1,0) Rand(1,1)

```

**Figure 3.** Local neighborhood definition.

but this is not a fixed value and the CD++ toolkit provides the means to adjust the value of  $d$  for specific instances where required.

- The **border** parameter refers to the edges of the cellular neighborhood and determines whether or not evaluation of cells will stop at an edge or will wrap around beyond an edge and pull data from the opposite edge. This parameter is related to the  $N$  parameter from the Cell-DEVS formalism with the border type enabling or restricting the contribution of specific cells to the set of input values ( $N$ ).

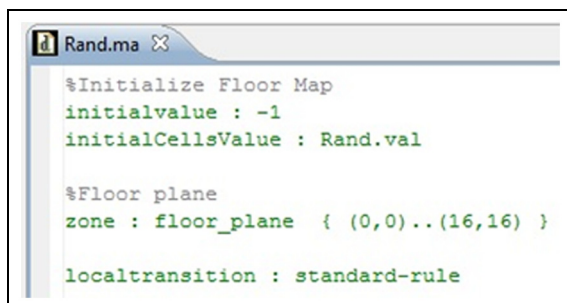
The next step is to establish the local neighborhood, the group of cells surrounding a target cell that will be used as inputs when evaluating the target cell. This neighborhood corresponds to the  $N$  parameter from the Cell-DEVS formalism. Figure 3 demonstrates the definition of the local neighborhood pattern for the Random Movement mode.

Different pieces of literature establish different types of local neighborhood patterns that have been determined through experimentation as effective for specific circumstances.<sup>6,17,18</sup> The (0, 0) cell refers to the target cell being evaluated, while the other cells detail their position relative to the target cell.

Then we establish the neighborhood initialization parameters that will determine either the initial states for every cell in the neighborhood or split up a neighborhood into zones if required or define the name of the rule-set that is going to govern the behavior of the model. These states belong to the parameter  $S$  from the Cell-DEVS formalism. Figure 4 demonstrates the definition of neighborhood initialization parameters for the Random Movement model.

The CD++ toolkit provides flexibility in the initialization of the local neighborhood. One can set every cell to a single value, define each cell one by one, define an entire row of cells, and even import text files that have predefined values for each cell. The different parameter types that allow developers this flexibility are defined in the CD++ User’s Guide.<sup>16</sup> For this research project, with the goal of dynamic models with minimal requirement to modify source files, importing text files with predefined values provides the best option. The **zone** parameter allows developers to partition a neighborhood and establish zone-specific rule-sets that apply only to cells residing in that zone. This can be a useful alternative to the coupling of





```

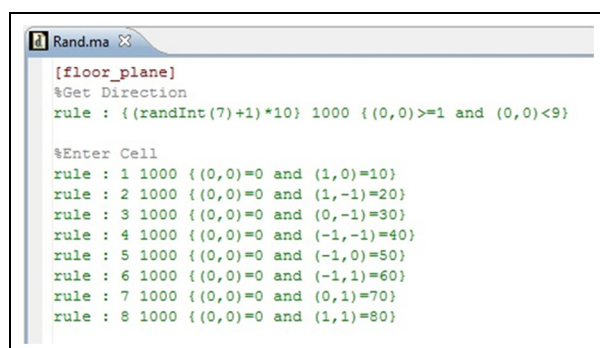
%Initialize Floor Map
initialvalue : -1
initialCellsValue : Rand.val

%Floor plane
zone : floor_plane { (0,0)..(16,16) }

localtransition : standard-rule

```

Figure 4. Neighborhood initialization parameters.



```

[floor_plane]
%Get Direction
rule : {(randInt(7)+1)*10} 1000 {(0,0)>=1 and (0,0)<9}

%Enter Cell
rule : 1 1000 {(0,0)=0 and (1,0)=10}
rule : 2 1000 {(0,0)=0 and (1,-1)=20}
rule : 3 1000 {(0,0)=0 and (0,-1)=30}
rule : 4 1000 {(0,0)=0 and (-1,-1)=40}
rule : 5 1000 {(0,0)=0 and (-1,0)=50}
rule : 6 1000 {(0,0)=0 and (-1,1)=60}
rule : 7 1000 {(0,0)=0 and (0,1)=70}
rule : 8 1000 {(0,0)=0 and (1,1)=80}

```

Figure 5. Partial random movement rule-set.

multiple neighborhoods and models. The **localtransition** parameter is a compulsory parameter, requiring developers to define a rule-set that will be applied to the whole model regardless of zones.

The final step for constructing a Cell-DEVS model is to establish the rules that will govern the behavior of the cells in the neighborhood. These rules must be constructed with a specific syntax, but there are no limits to how many rules one may use to evaluate each cell in a neighborhood for each periodic delay time. These rules correspond to the parameter  $\tau$  from the Cell-DEVS formalism. Figure 5 demonstrates the definition for part of a rule-set governing the Random Movement model.

From Figure 5, it can be seen that the first step is to identify the rule-set that the following rules will belong to. The next step is to write down the rules, starting with the **rule** identifier, then followed by three numerical components. The first component defines what the new state of the cell will be if it meets the requirements of the rule. The second component defines the delay time, which, for the transfer delays being used in this project, is the time it takes for a cell to change to its new state being evaluated as meeting the requirements of the rule. The final component is the expression that must be satisfied for a change of state to occur.

The first rule from Figure 5 will set the state of an evaluated cell to 10 multiplied by a random integer between

one and eight, after one second of transport delay, provided that the cell being evaluated is greater than or equal to one, and less than nine. The second rule from Figure 5 will set the state of an evaluated cell to one, after one second of transport delay, provided that the cell being evaluated is equal to zero, and that the cell to the immediate south is equal to 10.

#### 4. Emergency evacuation modeling

There is a large quantity of literature available exploring the mechanics behind egress and evacuation during an 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 available that has reviewed many of these products to determine the applicability of each model. This 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 et al.<sup>8</sup> 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 project 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. In their review of the commercial egress and evacuation models, Kuligowski et al.<sup>8</sup> noted that all egress and evacuation models were either movement focused, behavior focused, or a partial hybrid of the two. This split was also reflected by Radianti et al.,<sup>7</sup> whose five model categorizations can be assigned as movement based, behavior based, or a hybrid of the two. That said, Radianti et al. focus on crowd-based mechanics, arguing that the movement and behavior of a crowd trying to evacuate has vastly different mechanics to individuals and small groups trying to evacuate. Below we summarize major crowd modeling factors that shaped our work.

**Table 1.** Movement and behavior categorizations.

Movement / Behavior	Description
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 does not know where an exit is, they will almost certainly follow the crowd. This type of behavior can often be the result of counterflow, 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. For this research project, random behavior and twice the movement speed is used to model the panic behavior.

#### 4.1. 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 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 searching 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.

#### 4.2. 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 he/she is going may have the movement pattern changed after interaction with an authority figure who is patrolling the area to be evacuated.

#### 4.3. Categorization

There are a number of common elements for the evacuation and egress models that need to be defined. Table 1 shows the categorizations used in our work. These categories have been constructed based on studies previously reported.<sup>6-8</sup>

### 5. Experiments

The experiments are divided into four sections; the first demonstrating a proof of concept for the basic singular models, the second demonstrating a proof of concept for integrated models, the third demonstrating a proof of concept for the introduction of complications and the ability of the models to handle these complications, and the fourth measuring the performance of these models in larger scale environments, such as schools, malls, or even airports.

It is important to note during the experimentation phase that each model and simulation was compiled and executed five times. The reason for multiple compilations 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. Given the fixed seed value

**Table 2.** Basic model legend.

State	Meaning	Color
-1	Wall or obstacle	Black
0	Free space	No Color
1-800	Occupant	Blue
900	Exit	Green

that CD++ utilizes, we decided to pick a moderate repetition number to ensure accuracy of the simulation results. The simulation trials demonstrated insignificant variations in memory consumption and computation time. These results were then averaged for each model.

### 5.1. Proof of concept and analysis of basic models

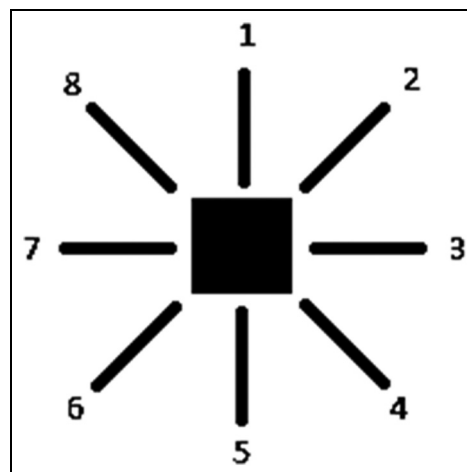
The purpose of this section is to produce a variety of small and simple evacuation models that will be built upon, expanded upon, and integrated incrementally. This section will start with five basic evacuation models, and through multiple iterations, it will incorporate elements of each basic evacuation model to produce more and more sophisticated evacuation scenarios.

**5.1.1. Basic evacuation models.** These evacuation models are designed to represent simple movements utilizing a single rule-set with no behavioral-driven changes of state. These models will be simulated using a small four-room building with a limited number of occupants. 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 800, depending on their movement type and the current direction they are moving in. Table 2 provides the basic model color/value scheme.

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, the direct relationship between rule-set and mechanic will only be explored in detail for the Core Concepts and the Random Movement model. For the remaining 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, which is available open-source from Egress Cell-DEVS Models Source Code.<sup>4</sup>

**5.1.2. Core concepts.** The development of Cell-DEVS models and the rules that govern them, in the context of this research project, are driven by the following core concepts.

- Cells come in four generic types – obstacles, empty space, exits, and occupants. Obstacles include

**Figure 6.** Compass detailing directional data.

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 to 800, with directional data embedded in the 1s and 10s, and movement type data embedded in the 100s.
- 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.

Figure 6 provides a visual representation of this 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 10. After a move in the chosen direction has been successfully completed, the state of the cell will be divided by 10. 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 10 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 10 indicate that the cell is ready to move in the desired direction.

- Given the period evaluation of each cell in the neighborhood, 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 neighborhood, allowing a cell to predict the movement of other cells more than two spaces away. When two cells wish 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 1 being the highest priority direction through to 8 being the lowest priority direction. For instance, if two agents are trying to occupy an empty cell in between, one agent will have a south direction (moving down, priority = 5) while the other agent will have a north direction (moving up, priority = 1), then according to their movement direction priority numbers explained in Figure 6, the agent with higher direction priority (moving down) will occupy the cell. Song et al.<sup>15</sup> provide more insights on collision detection and avoidance in cellular spaces.
- The preferred neighborhood shape for evaluating each cell on the zero-plane will be a  $5 \times 5$  grid with the cell to be analyzed residing at the center. For cells residing on other planes, the neighborhoods defining how they are to be analyzed, 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.

**5.1.3. 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 is a very simple mechanic that will play a part in future models and simulations that involve panic states.

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

- **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 such as Get Direction. The rule that defines this movement is as follows:

- rule : 0 1000  $\{(0,0) > 0 \text{ and } (0,0) < 90 \text{ and } ((0,1)=900 \text{ or } (1,1)=900 \text{ or } (1,0)=900 \text{ or } (1,-1)=900 \text{ or } (0,-1)=900 \text{ or } (-1,-1)=900 \text{ or } (-1,0)=900 \text{ or } (-1,1)=900)\}$ ;

- **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 1 and 8. The value of the integer represents the direction of movement, as defined by Figure 6. The rule that defines this movement is as follows:

- rule :  $\{(\text{randInt}(7) + 1) * 10\}$  1000  $\{(0,0) \geq 1 \text{ and } (0,0) < 9\}$ ;

- **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. The rules that define this movement are as follows:

- rule : 1 1000  $\{(0,0)=0 \text{ and } (1,0)=10\}$ ;
- rule : 2 1000  $\{(0,0)=0 \text{ and } (1,-1)=20 \text{ and } (1,0)!=10\}$ ;
- rule : 3 1000  $\{(0,0)=0 \text{ and } (0,-1)=30 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20)\}$ ;
- rule : 4 1000  $\{(0,0)=0 \text{ and } (-1,-1)=40 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20 \text{ or } (0,-1)!=30)\}$ ;
- rule : 5 1000  $\{(0,0)=0 \text{ and } (-1,0)=50 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20 \text{ or } (0,-1)!=30 \text{ or } (-1,-1)!=40)\}$ ;
- rule : 6 1000  $\{(0,0)=0 \text{ and } (-1,1)=60 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20 \text{ or } (0,-1)!=30 \text{ or } (-1,-1)!=40 \text{ or } (-1,0)!=50)\}$ ;
- rule : 7 1000  $\{(0,0)=0 \text{ and } (0,1)=70 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20 \text{ or } (0,-1)!=30 \text{ or } (-1,-1)!=40 \text{ or } (-1,0)!=50 \text{ or } (-1,1)!=60)\}$ ;
- rule : 8 1000  $\{(0,0)=0 \text{ and } (1,1)=80 \text{ and } ((1,0)!=10 \text{ or } (1,-1)!=20 \text{ or } (0,-1)!=30 \text{ or } (-1,-1)!=40 \text{ or } (-1,0)!=50 \text{ or } (-1,1)!=60 \text{ or } (0,1)!=70)\}$ ;

- **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  $5 \times 5$  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. The rules that define this movement are as follows:

- rule : 0 1000  $\{((0,0)=10 \text{ and } (-1,0)=0)\}$ ;
- rule : 0 1000  $\{((0,0)=20 \text{ and } (-1,1)=0 \text{ and } (0,1)!=10)\}$ ;



- rule : 0 1000  $\{((0,0)=30 \text{ and } (0,1)=0 \text{ and } (1,1)!=10 \text{ and } (1,0)!=20)\}$ ;
- rule : 0 1000  $\{((0,0)=40 \text{ and } (1,1)=0 \text{ and } (2,1)!=10 \text{ and } (2,0)!=20 \text{ and } (1,0)!=30)\}$ ;
- rule : 0 1000  $\{((0,0)=50 \text{ and } (1,0)=0 \text{ and } (2,0)!=10 \text{ and } (2,-1)!=20 \text{ and } (1,-1)!=30 \text{ and } (0,-1)!=40)\}$ ;
- rule : 0 1000  $\{((0,0)=60 \text{ and } (1,-1)=0 \text{ and } (2,-1)!=10 \text{ and } (2,-2)!=20 \text{ and } (1,-2)!=30 \text{ and } (0,-2)!=40 \text{ and } (0,-1)!=50)\}$ ;
- rule : 0 1000  $\{((0,0)=70 \text{ and } (0,-1)=0 \text{ and } (1,-1)!=10 \text{ and } (1,-2)!=20 \text{ and } (0,-2)!=30 \text{ and } (-1,-2)!=40 \text{ and } (-1,-1)!=50 \text{ and } (-1,0)!=60)\}$ ;
- rule : 0 1000  $\{((0,0)=80 \text{ and } (-1,-1)=0 \text{ and } (0,-1)!=10 \text{ and } (0,-2)!=20 \text{ and } (-1,-2)!=30 \text{ and } (-2,-2)!=40 \text{ and } (-2,-1)!=50 \text{ and } (-2,0)!=60 \text{ and } (-1,0)!=70)\}$ .
- **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 1, allowing the Get Direction mechanic to set a new random direction. The rules that define this movement are as follows:
  - rule : 1 1000  $\{((0,0)=10 \text{ and } (-1,0)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=20 \text{ and } (-1,1)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=30 \text{ and } (0,1)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=40 \text{ and } (1,1)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=50 \text{ and } (1,0)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=60 \text{ and } (1,-1)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=70 \text{ and } (0,-1)!=0)\}$ ;
  - rule : 1 1000  $\{((0,0)=80 \text{ and } (-1,-1)!=0)\}$ .

**5.1.4. 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, 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  $3 \times 3$  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 as follows.

- **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 are 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 in which 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.

**5.1.5. 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  $3 \times 3$  grid. All five mechanics are maintained from the Optimal Movement model, with the Get Direction and Maintain Current Direction mechanics slightly modified as follows.

- **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 Figure 6, 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.

**5.1.6. Patrol Movement Model.** In this evacuation model, a person of authority will follow an established path, patrolling the building to determine whether or not everyone has been evacuated. In this model there are two planes, a plane

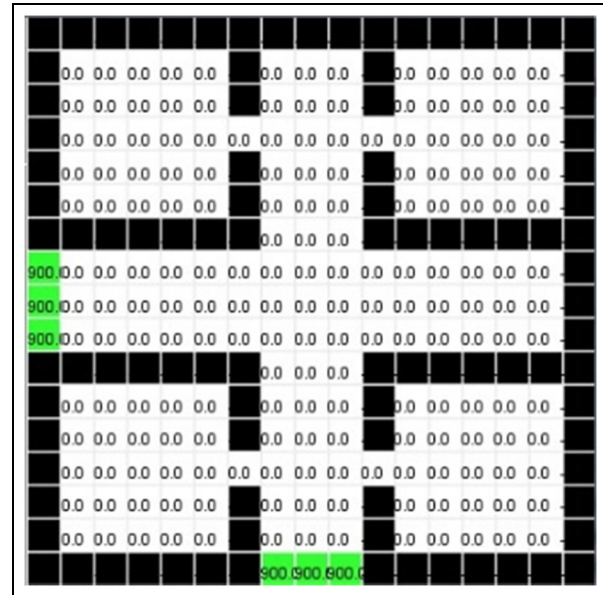
that contains the floor map and the occupant positions, and a plane that contains the route for the person of authority to follow during their patrol. For this basic model the interactions with other occupants will not be handled; this is just a proof of concept to show that occupants can follow a patrol path. There is a variation on this 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  $3 \times 3$  grid. The Enter Cell, Leave Cell, and Exit Area mechanics are maintained, while the Get Direction mechanic is modified as follows.

- **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 in which 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 Figure 7, there were 12 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.

**5.1.7. 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



**Figure 7.** Basic simulation floor map (white cells = 0.0, green cells (exit areas) = 900). (Color online only)

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 13 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

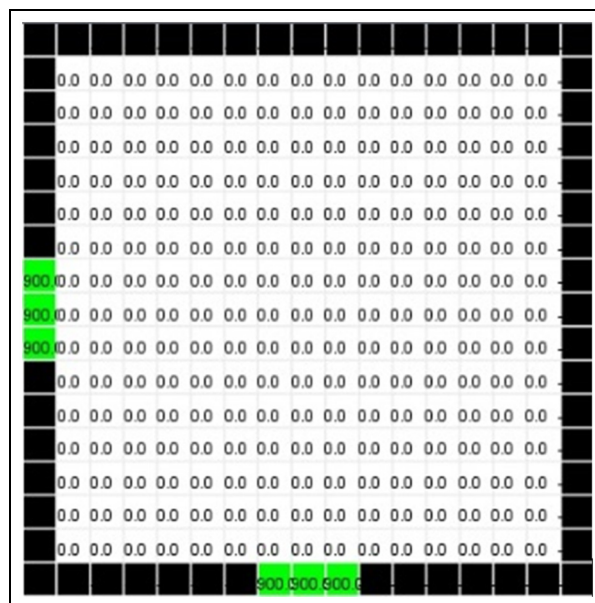
leading occupant to change direction in an anticlockwise manner until there is no observable wall blocking the path forward.

- **Leader: Keep Current Direction** – this mechanic evaluates whether a leading occupant can maintain 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 the 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.

Finally, to simplify the mechanics of this model, instead of using the floor map provided in Figure 7, a new floor map with the walls removed will be used as detailed in Figure 8.

## 5.2. Proof of concept and analysis of integrated models

The purpose of this section is to explore the integration of the models established in Section 1, demonstrating how



**Figure 8.** Follow the herd floor map (white cells = 0.0, green cells (exit areas) = 900). (Color online only.)

these models can be pulled together to produce more intricate models that better reflect how egress and evacuation may play out.

**5.2.1. Patrol Finding Lost People.** In this evacuation model, the Patrol Movement Model, Random Movement Model and Directional Model have been merged so that a Patrolling Occupant is able to provide directional information to any occupants found while traversing their patrol route. Similar to the Follow the Herd (FtH) Movement Model, there are a variety of occupant types represented by this model:

- 101–180 – Random Moving Occupant – Orange;
- 201–280 – Informed Occupant – Purple;
- 301–380 – Patrolling Occupant – Blue.

It is also important to note that this is the first model to make use of three planes, with the zero plane representing the floor map, the first plane representing the patrol route for the patrolling occupants, and the second plane representing the directional information for informed occupants.

This model makes use of 13 mechanics defined previously by the Patrol Movement, Random Movement, and Directional Movement Models, modifying them slightly so that the Patrol Movement mechanics only apply to the patrolling occupants, the Random Movement mechanics only apply to random moving occupants, and the Directional Movement mechanics only apply to the Informed Occupants. The following new mechanic is introduced by this model.

- **Transition from Random Moving to Informed Occupant** – this mechanic evaluates the neighborhood surrounding a random moving occupant, and if a patrolling occupant is observed within this neighborhood, the random moving occupant will become an informed occupant and start accessing the directional information plane.

**5.2.2. Follow the Herd – Collecting Random Moving People.** In this evacuation model, the Random Movement Model and FtH Model have been merged so that the herd is forced to deal with random moving occupants rather than just stationary occupants. Below are the occupant types represented by this model:

- 001–080 – Random Moving Occupant – Red;
- 110–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue.

This model makes use of 18 mechanics defined previously by the Random Movement and FtH Movement Models, modifying them slightly so that the Random Movement mechanics only apply to the random moving occupants, and the FtH Movement mechanics only apply to the leading and following occupants. The Transition from Stationary to Following Occupant mechanic from the FtH Movement Model has been modified so that it can now transition random moving occupants to following occupants.

**5.2.3. Follow the Herd – New Leaders Emerge.** In this evacuation model, the previous FtH – Collecting Random Moving People Model has been adjusted, allowing for the creation of new leaders and multiple herds in the same area. It also acts to merge a herd when two leaders come together. Below are the occupant types represented by this model:

- 001–080 – Random Moving Occupant – Red;
- 110–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue.

This model makes use of the 18 mechanics defined previously by the FtH – Collecting Random Moving People Model, and introduces two new mechanics as follows.

- **Transition from Random Moving to Leading Occupant** – this mechanic evaluates the neighborhood surrounding a random moving occupant, searching for another random moving occupant. If two random moving occupants meet, the most northern or eastern occupant will be transitioned into a leading occupant.

- **Merging of Herds** – this mechanic evaluates the neighborhood surrounding a leading occupant, searching for another leading occupant. If two leading occupants meet, the most northern or eastern occupant will remain as a leading occupant, while the most southern or western occupant will be transitioned into a following occupant.

**5.2.4. Follow the Herd – Herd Given Directional Information.** In this evacuation model, the previous FtH – New Leaders Emerge Model has been adjusted, allowing for herds to absorb directional information provided by an occupant who has access to directional information. Below are the occupant types represented by this model:

- 001–080 – Random Moving Occupant – Red;
- 101–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue;
- 301–380 – Directional Occupant – Dark Purple;
- 401–480 – Informed Herd Occupant – Light Purple.

This model makes use of the 25 mechanics defined previously by the FtH – New Leaders Emerge Model and the Directional Movement Model, modifying them slightly so that the FtH mechanics apply to the random moving occupants, following occupants, and leading occupants, and the Directional Movement mechanics apply to the directional occupants and following occupants. This model also introduces the following mechanic.

- **Transition from Herd to Informed Herd Occupant** – this mechanic evaluates the neighborhood surrounding a leading or following occupant, searching for a directional occupant. If a directional occupant is within the neighborhood, the leading or following occupant is transitioned to an informed herd occupant and can start accessing directional information from the directional plane. This mechanic is extended by having leading or following occupants transition to informed herd occupants if there are other informed herd occupants in their neighborhood. This is based on the idea that information passes fast and freely through a herd.

### 5.3. Proof of concept and analysis of integrated models with complications

The purpose of this section is to explore how an integrated model will respond to different types of complications, demonstrating the responsiveness and reactivity of the integrated models. In this section we will look at a single integrated model and explore how it responds to these complications.



**5.3.1. Panic.** As the duration of the evacuation increases, occupants become more agitated, and will start to behave more and more erratically. Occupants may start to run, or move in seemingly random behaviors, and unless they can be calmed their erratic behavior will continue to grow.

In this evacuation model, the previous FtH – Herd Given Directional Information Model has been adjusted, with occupants in a prolonged stationary state transitioning to a panicked state. Below are the occupant types represented by this model:

- 001–080 – Random Moving / Panicked Occupant – Red;
- 101–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue;
- 301–380 – Directional Occupant – Dark Purple;
- 401–480 – Informed Herd Occupant – Light Purple.

For the purpose of this model, it is assumed that random moving occupants exhibit equivalent behavior to panicked occupants. Future iterations of this suite of models should consider developing a more sophisticated panic response. The floor map provided by Figure 8 was modified so that the western exit had only one exit cell, and required going through a bottle-neck to get to the exit. This was done to increase the likelihood of having stationary occupants. This model makes use of the 25 mechanics defined previously by the FtH – Herd Given Directional Information Model. This model also introduces the concept of a panic plane, one that can track how long an occupant has remained in a stationary position. It also introduces the following mechanic.

- **Trigger Panic** – this mechanic evaluates an occupant cell by storing its state value in the corresponding cell on the panic plane. In the next time step, it compares the state value of the same cell with the corresponding cell on the panic plane, and if the values are the same then the state value of the panic plane cell is multiplied by two, indicating that the occupant has been stationary for two time steps. This process continues until the panic plane cell state value is equal to the occupant cell state value multiplied by five, at which point a state change is triggered, turning the stationary occupant into a panicked occupant. All occupants defined in this model are subject to this mechanic.

**5.3.2 Blocked exit.** In this complication, occupants following directional- or herd-based movements will be confronted by a blocked exit and will have to make a decision. In this model, an authority figure will be placed at the blocked exit, and will provide all occupants with alternative directions to the nearest unblocked exit.

In this evacuation model, the previous Panic Model has been adjusted, with authoritative occupants now able to provide all other occupants with alternative direction information stored on a new information plane. Below are the occupant types represented by this model:

- 001–080 – Random Moving / Panicked Occupant – Red;
- 101–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue;
- 301–380 – Directional Occupant – Dark Purple;
- 401–480 – Informed Herd Occupant – Light Purple;
- 500–580 – Alternate Path Occupant – Light Blue;
- 1001 – Authoritative Occupant – Gray.

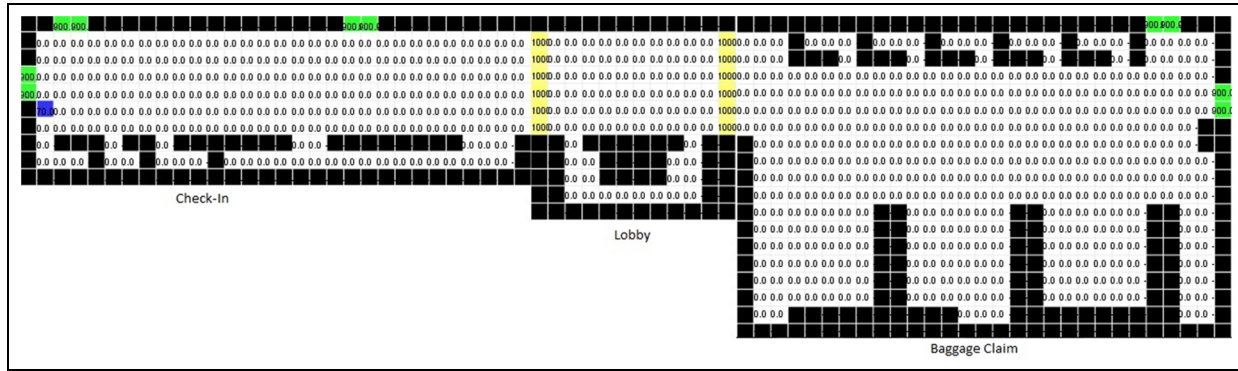
This model makes use of the 27 mechanics defined previously by the Panic Model. This model also introduces the concept of an alternative direction information plane, one that provides direction information to the nearest unblocked exit. This means that another five Directional Movement Model mechanics need to be added to this model, tuned specifically for accessing the alternative direction information plane. It also introduces the following mechanic.

- **Transition to Alternate Path Occupant** – this mechanic evaluates an occupant cell, searching for an authoritative occupant. If an authoritative occupant is discovered within their neighborhood, the occupant will transition to an Alternate Path Occupant and begin progress towards the nearest unblocked exit.

**5.3.3 Crowd control.** In this complication, a patrolling occupant is introduced to provide specific directional information to any occupants that they come in contact with. For the purposes of this model, patrolling occupants will transition all other occupants to an Alternate Path Occupant, but future iterations of this model suite should look at implementing multiple alternative direction information planes with patrolling occupants given the freedom to choose which alternate path applies, depending on the current situation and location.

In this evacuation model, the previous Blocked Exit Model has been adjusted, with patrolling occupants now able to provide all other occupants with alternative direction information stored on a new information plane. Below are the occupant types represented by this model:

- 001–080 – Random Moving / Panicked Occupant – Red;
- 101–180 – Following Occupant – Yellow;
- 201–280 – Leading Occupant – Blue;
- 301–380 – Directional Occupant – Dark Purple;



**Figure 9.** Cellular representation of the first-level terminal layout. (Color online only.)

- 401–480 – Informed Herd Occupant – Light Purple;
- 500–580 – Alternate Path Occupant – Light Blue;
- 1001 – Authoritative Occupant – Gray;
- 1101–1180 – Patrolling Occupant – Gray.

This model makes use of the 33 mechanics defined previously by the Blocked Exit Model. This model also has to take into account the patrol route plane for the patrolling occupants, meaning that there are now five separate planes containing specific information. The four Patrol Movement mechanics need to be added to this model, tuned specifically for the new state values of the patrolling occupant. It also needs to implement a second Transition to Alternate Path Occupant mechanic tuned to respond to patrolling occupants. This makes up a total of 38 movement mechanics controlling the various occupants that make up this model.

#### 5.4 Cell-DEVS representation of an airport area

In this section, we take the models and complications we have examined in the previous three sections, and apply them to a more realistic scenario. For this project, the scenario will be the evacuation of a building modeled after the Daytona Beach Airport. In this section there will be more sophisticated measurements, comparing and contrasting models for the time it takes to evacuate, how many people go through each exit, and identifying any potential bottle-necks that slow down the evacuation.

The Daytona Beach Airport is divided into three main areas – the downstairs terminal, the upstairs terminal and security, and the concourse. Given the shape and size of these areas, the space on the first-level terminal was divided into three main areas – baggage claim, check-in, and lobby. Figure 9 provides a comparison between the Terminal Layout provided by the official Daytona Beach Airport Website (<http://www.flydaytonafirst.com/>) and the approximation provided by the cellular representation.

When comparing Figures 9 and 10, it can be seen that there are a number of differences between the two images

as a result of design decisions. The first design decision was to remove the angular aspect of the terminal design by stitching together three separate cellular designs of the terminal. The Check-In, Lobby, and Baggage Claim each represent separate components within the top-level Cell-DEVS model, which have been coupled together at the seams represented by yellow colored cells with a state value of 1000. The second design decision was to remove the areas that are not accessible by the general public. The third design decision was to get the cellular representation as close to scale as possible, and only modify the design when reaching the limits of the Cell-DEVS visualizer. During the development of the concourse component of the Daytona Beach Airport, it became apparent that even a significantly scaled representation would be far too large for the CD++ visualization tool to handle. The model could still be built and the simulation would run without fault, but visualization tool provides important contextual information that would be extremely difficult to pull out from thousands of lines of text in a log file. For this reason, development of the concourse component was not completed; however, future work could examine alternative methods for building a model of the concourse, possibly by dividing the area into smaller areas, and coupling these smaller areas together. When building the overall model, a single source file was created with four sub models representing the Check-In, Lobby, Baggage, and Plaza. Individually, these four models made use of the Crowd Control Model established in the previous section, the concept being that the Crowd Control rule-set should be independent of the information plane declarations. The Crowd Control rule-set was copied and pasted without modification into these four models and, individually, these four models could be successfully simulated.

Looking at the bigger picture, these models were coupled using the CD++ coupling functionality, allowing for the transfer of occupants from model to model through the use of dedicated event ports. With the models coupled, and no occupants defined in the airport, the overarching model simulated without issue. When introducing

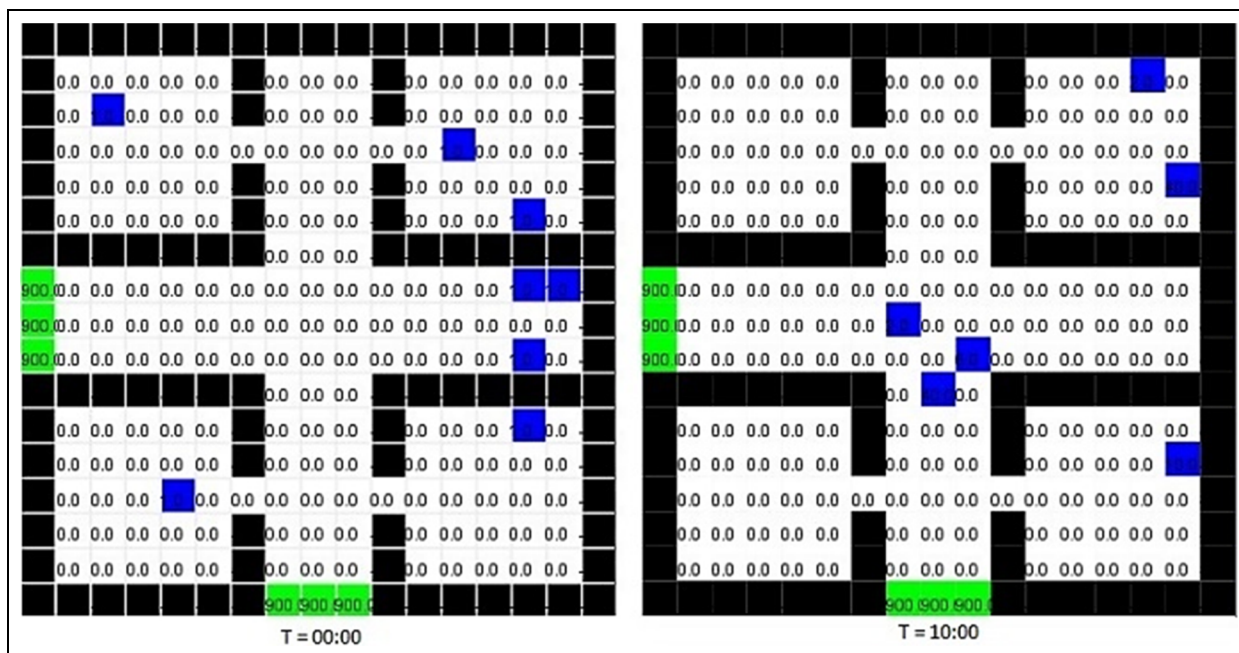


Figure 10. Random Movement Model visualization.

occupants into the Check-In sub model, the overarching model simulated without issue. When introducing occupants to any other sub model, the overarching model would fail to simulate, the compiler citing neighborhood issues without specifically pointing to the cause of the error. After extensive bug checking using a half split methodology, the solution to the problem was to reduce the number of couplings between models and implement a priority-based system to avoid simultaneous event handling.

## 6. Results and analysis

The following sections detail the results and analysis of the simulations performed based on the models constructed. These sections parallel the same sections from the methodology component, and will 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. Please note that the simulation time reported for each model represents the total real-time that it took all pedestrians in that model to evacuate the area. As explained in Section 5.1.2, each move (moving from one cell to another cell) represents one second in real-time.

### 6.1. Proof of concept results

6.1.1. *Random Movement Model.* The Random Movement model source file was 76 LOCs on completion, with 28 of those lines represented as Cell-DEVS rules. This

Table 3. Random Movement Model simulations.

Simulation time	Computation time	Memory usage
35:55 minutes	18.54 seconds	23.5 MB

represents the smallest model size in terms of LOCs and rules, which was to be expected given that this was the least complex model constructed. Table 3 records the results of the simulation execution for the Random Movement Model. Computation time was measured using a stopwatch and may be subject to human error. Memory usage was recorded using the Windows Resource Monitor tool.

For the Random Movement Model, it takes 35:55 minutes (or 2155 simulated steps) for eight occupants to escape the structure defined by Figure 6. Bearing in mind that the Random Movement Model makes use of the Random functionality provided by CD++ , it can be seen from this table that the CD++ seed is singular, even for its random number generation. Performing a character-by-character comparison of the log files shows that they are identical. Figure 10 shows snapshots at regular intervals of the Random Movement Model completing its simulation. It shows that with random movement, the occupants found it difficult to find that one door leading out of a room, but once they were out of their room, the narrow hallways and large exits meant that there was a greater probability of the occupant travelling in a direction that would lead to an exit.



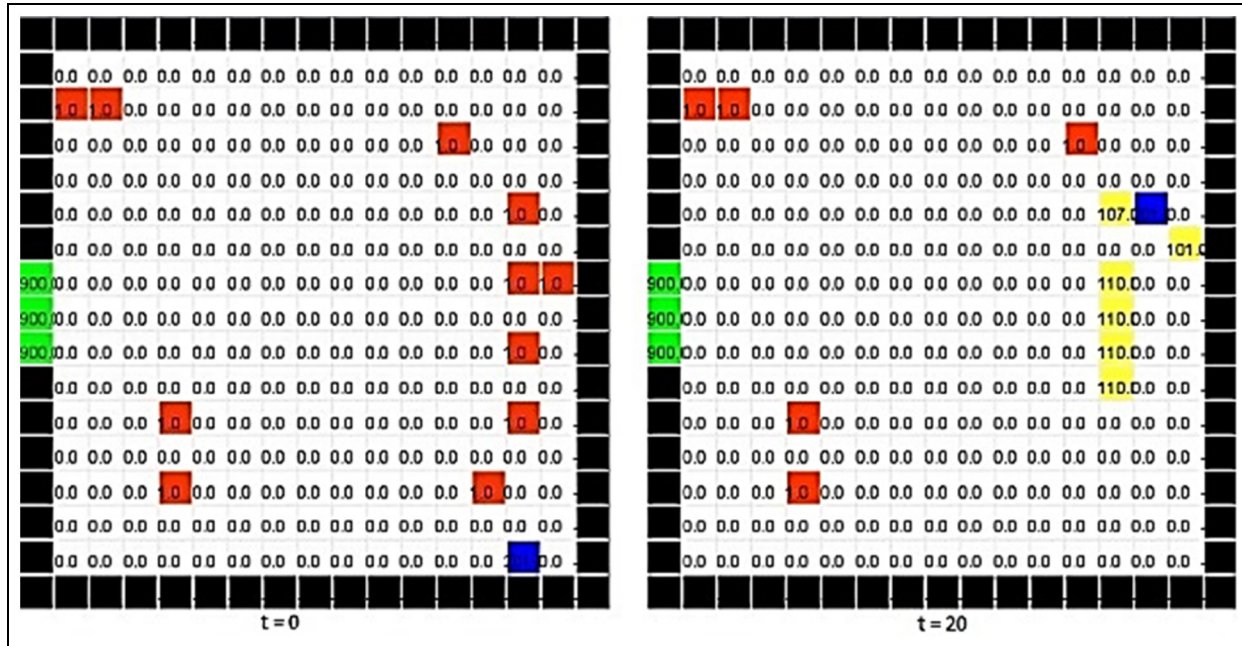


Figure 11. Follow the Herd Movement Model visualization.

Table 4. Optimal Movement Model simulation.

Simulation time	Computation time	Memory usage
27 seconds	1.77 seconds	7 MB

Table 5. Directional Movement Model simulation.

Simulation time	Computation time	Memory usage
34 seconds	2.51 seconds	10 MB

6.1.2 *Optimal Movement Model.* The Optimal Movement Model source file was 95 LOCs on completion, with 35 of those lines represented as Cell-DEVS rules. This represents a small increase in size and complexity from the Random Movement Model, which was to be expected given that this model had to define and start using multiple planes. Table 4 records the results of the simulation execution for the Optimal Movement Model and presents the average values for five trials of the simulation. For the Optimal Movement Model, it takes 27 seconds (or 27 simulated steps) for nine occupants to escape the structure defined by Figure 6. This shows a dramatic and expected increase from the Random Movement Model, with all occupants choosing their shortest route to the exit. Despite navigating the area in the shortest timeframe, this is an idealistic type of model rarely representative of a real-life situation, and should be treated as idealistic rather than realistic.

6.1.3 *Directional Movement Model.* The Directional Movement Model source file was 93 LOCs on completion, with 35 of those lines represented as Cell-DEVS rules. This represents a slight increase in LOCs, which could be comment related, and no increase in rule size. This was to

be expected as the Directional Movement Model and Optimal Movement Model both do almost the same thing, just with different data. Table 5 records the results of the simulation execution for the Directional Movement Model. For the Directional Movement Model, it takes 34 seconds (or 34 simulated steps) for nine occupants to escape the structure defined by Figure 6. Despite having the same layout and occupant position, this is longer than for the Optimal Movement Model by seven seconds. This is because the Optimal Model is optimized for shortest path to the exit, while the Directional Movement Model only gives an indicative direction. Despite navigating the area in the shortest timeframe, this is an idealistic type of model rarely representative of a real-life situation, and should be treated as idealistic rather than realistic.

6.1.4 *Patrol Movement Model.* The Patrol Movement Model source file was 97 LOCs on completion, with 39 of those lines represented as Cell-DEVS rules. This represents a small increase in size and complexity, which is expected given that this model had to include additional rules to handle patrol route crossover. Table 6 records the results of the simulation execution for the Patrol Movement



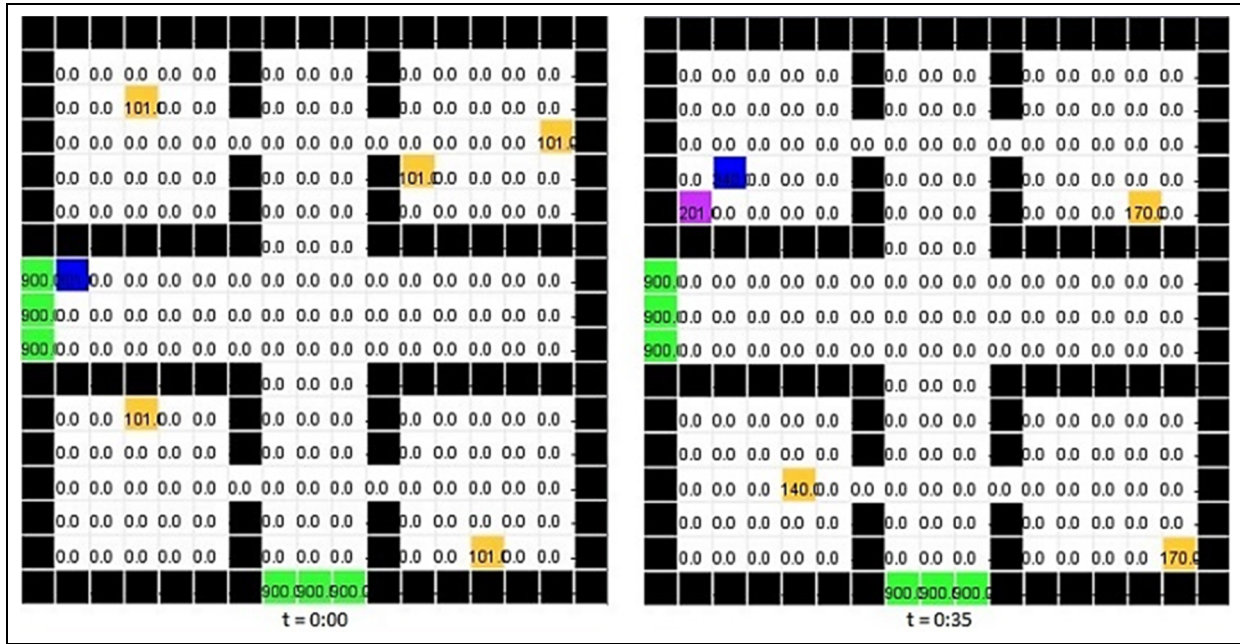


Figure 12. Patrolling for lost people visualization.

Table 6. Patrol Movement Model simulation.

Simulation time	Computation time	Memory usage
2:58 minutes	2.04 seconds	8 MB

Table 7. Follow the Herd Movement Model simulation.

Simulation time	Computation time	Memory usage
1:16 minutes	4.11 seconds	12.33 MB

Model. For the Patrol Movement Model, it takes 2:58 minutes (or 178 simulated steps) for the patrolling occupant to complete their patrol route of Figure 6. This is a fixed time, depended entirely on the route planned for the patrolling occupant.

**6.1.5 Follow the Herd Movement Model.** The FtH Movement Model source file was 207 LOCs on completion, with 116 of those lines represented as Cell-DEVS rules. This represents a significant increase in size and complexity of the model compared to the previously discussed ones. This is driven by the fact that multiple occupant types and 13 different movement mechanics are utilized in this model. Table 7 records the results of the simulation execution for the FtH Movement Model. For the FtH Movement Model, it takes 1:16 minutes (or 76 simulated steps) for 10 occupants to escape the structure defined by Figure 8. Given

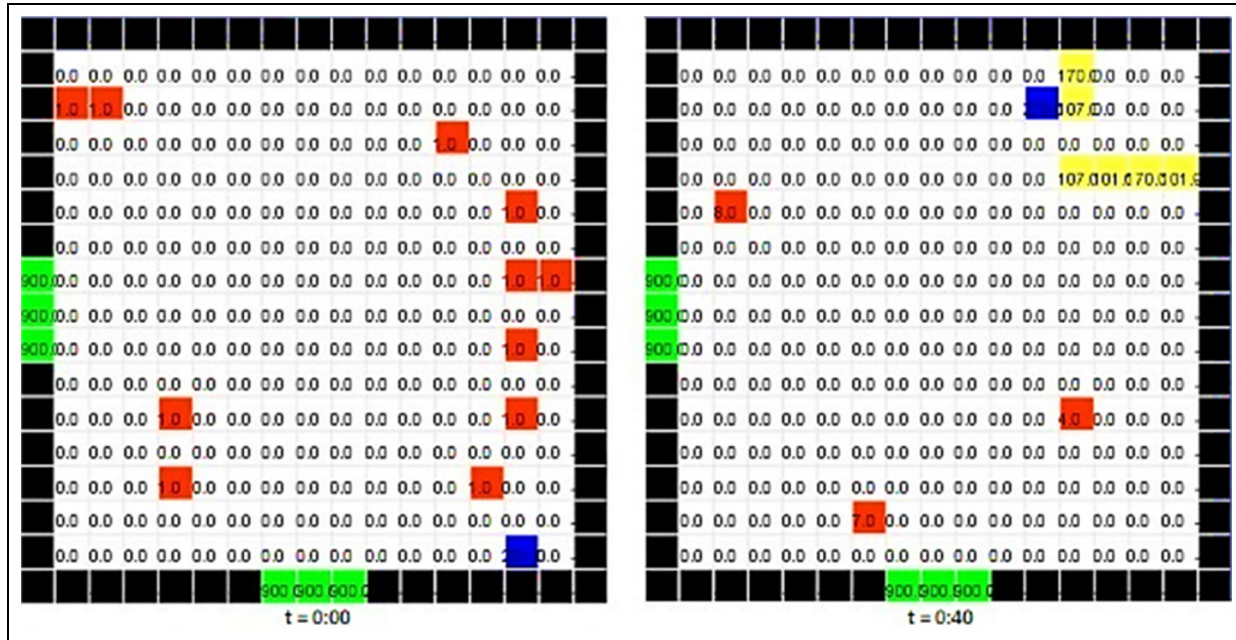
Table 8. Patrolling for Lost People Simulation.

Simulation time	Computation time	Memory usage
3:03 minutes	10.48 seconds	16 MB

that the FtH Model uses stationary occupants, and that the leading occupant did not follow a path that covered the entire area, two of the stationary occupants were not made into following occupants and were not able to escape. Figure 11 shows snapshots of the FtH Model. It shows the path the leading occupant took in navigating to the exit, and demonstrated how stationary occupants joined the herd, and how following occupants react to the leading occupant and other following occupants.

## 6.2 Integration results

**6.2.1 Patrol Finding Lost People.** The Patrol Finding Lost People Model source file was 203 LOCs on completion, with 105 of those lines represented as Cell-DEVS rules. This was to be expected given that it integrates three of the previous models – the Random Movement Model, the Patrol Movement Model, and the Directional Movement Model. Table 8 records the results of the simulation execution for the Patrol Finding Lost People Model. For the Patrol Finding Lost People Model, it takes 3:03 minutes (or 183 simulated steps) for five occupants to escape the structure defined by Figure 6. This was largely predefined given the patrol route; however, it took five steps longer than the Patrol Movement Model. This is because there



**Figure 13.** Follow the Herd Collecting Random Movers visualization.

are now random moving occupants interrupting the path of the patrolling occupant. Figure 12 shows snapshots of the Patrol Finding Lost People Movement Model completing its simulation. It shows the patrolling occupant informing each of the random moving occupants about the location of the exit, and shows the newly informed occupants moving towards that exit.

**6.2.2. Follow the Herd – Collecting Random Moving People.** The FtH – Collecting Random Moving People Model source file was 259 LOCs on completion, with 154 of those lines represented as Cell-DEVS rules. This is expected as it is a direct iteration of the FtH model, integrating the Random Movement Model. Table 9 records the results of the simulation execution for the FtH – Collecting Random Moving People Model. For the FtH – Collecting Random Moving People Model, it takes 3:19 minutes (or 199 simulated steps) for 12 occupants to escape the structure defined by Figure 8. This is an improvement from the previous FtH model in that all occupants escaped, but the time was inflated because of one random moving occupant being stuck in a corner. Figure 13 shows snapshots of the FtH model. It was observed that the leading occupant did not pick up as many following occupants as the previous model, but this was to be expected as the occupants were no longer stationary.

**6.2.3 Follow the Herd – New Leaders Emerge.** The FtH – New Leaders Emerge Model source file was 311 LOCs on

**Table 9.** Follow the Herd Collecting Random Movers simulation.

Simulation time	Computation time	Memory usage
3:19 minutes	10.34 seconds	21 MB

**Table 10.** Follow the Herd with New Leaders Emerging simulation.

Simulation time	Computation time	Memory usage
1:39 minutes	8.27 seconds	15 MB

completion, with 181 of those lines represented as Cell-DEVS rules. This represents the next iteration of the FtH model, with movement mechanics now handling the emergence new leaders and herds, and the merging of multiple herds. Table 10 records the results of the simulation execution for the FtH – New Leaders Emerge. For the FtH – New Leaders Emerge, it takes 1:39 minutes (or 99 simulated steps) for 12 occupants to escape the structure defined by Figure 8. This shows an improved performance, with more herds helping to reduce the number of single random movement occupants from escaping into corners.

**6.2.4. Follow the Herd – Herd Given Direction Information.** The FtH – Herd Given Direction Information

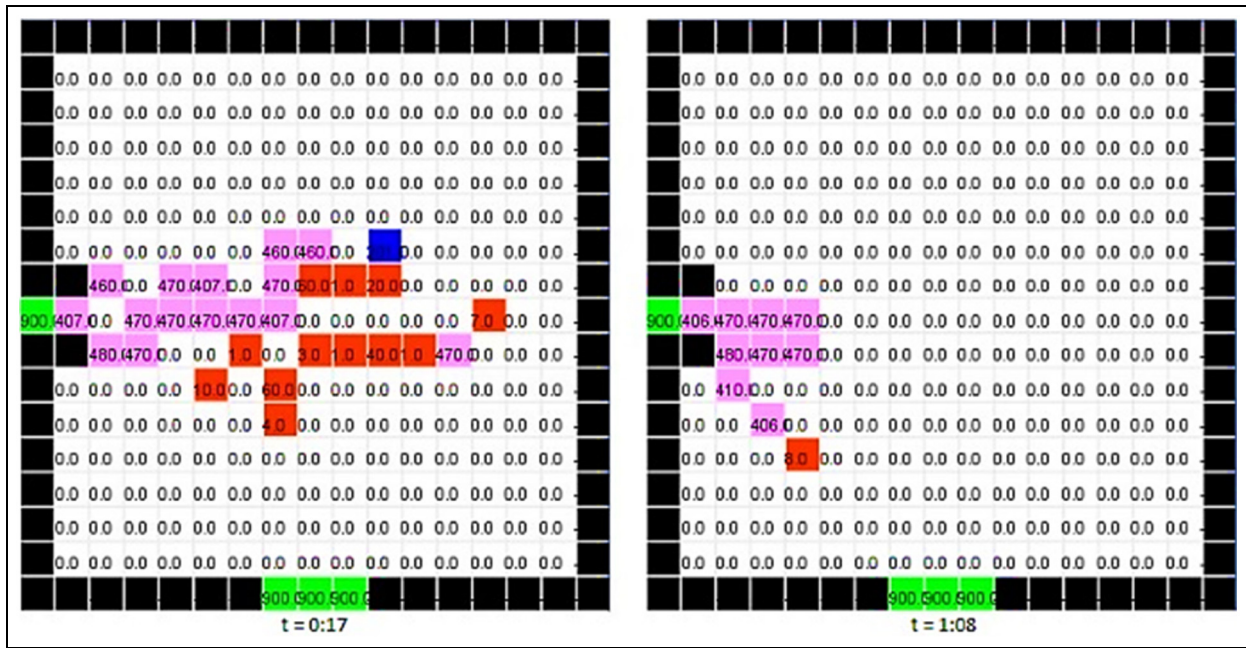


Figure 14. Panic visualization.

Table 11. Follow the Herd Given Directional Information simulation.

Simulation time	Computation time	Memory usage
45 seconds	5.70 seconds	17 MB

Model source file was 418 LOCs on completion, with 269 of those lines represented as Cell-DEVS rules. This is the next FtH iteration, and sees a sizeable jump in integrated functionality. Table 11 records the results of the simulation execution for the FtH – Herd Given Directional Information Model. For the FtH – Herd Given Directional Information Model, it takes 45 seconds (or 45 simulated steps) for eight occupants to escape the structure defined by Figure 8. This is a sizeable decrease; however, this simulation was designed to show the impact of directional occupants on herd occupants, and the positioning of these occupants was tailored for this demonstration.

### 6.3 Complication results

6.3.1 *Panic.* The Panic Model source file was 456 LOCs on completion, with 278 of those lines represented as Cell-DEVS rules. This is an iteration of the final FtH model, introducing panic mechanics to the modeling and simulation suite. Table 11 records the results of the simulation execution for the Panic Model. For the Panic Model, it takes 2:35 minutes (or 155 simulated steps) for 33 occupants to escape the modified Figure 8 structure. The large increase in occupants, along with introduction of a bottle-

Table 12. Panic simulation.

Simulation time	Computation time	Memory usage
2:35 minutes	18.80 seconds	26 MB

Table 13. Blocked Exit simulation.

Simulation time	Computation time	Memory usage
6:19 minutes	33.20 seconds	28 MB

neck to the floor map, was designed to maximize the chance of occupants entering a panicked state. Figure 14 shows snapshots of the Panic Model completing its simulation. It shows multiple occupants entering a panicked state, before calming down through the influence of other occupants.

6.3.2 *Blocked exit.* The Blocked Exit Model source file was 517 LOCs on completion, with 321 of those lines represented as Cell-DEVS rules. This represents an increment of the Panic Model, with the introduction of authority figures who can guide occupants along alternative routes. Table 13 records the results of the simulation execution for the Blocked Exit Model. For the Blocked Exit Model, it takes 6:19 minutes (or 379 simulated steps) for 33 occupants to escape the modified Figure 8 structure. This is a large increase, caused by an isolated panicked occupant



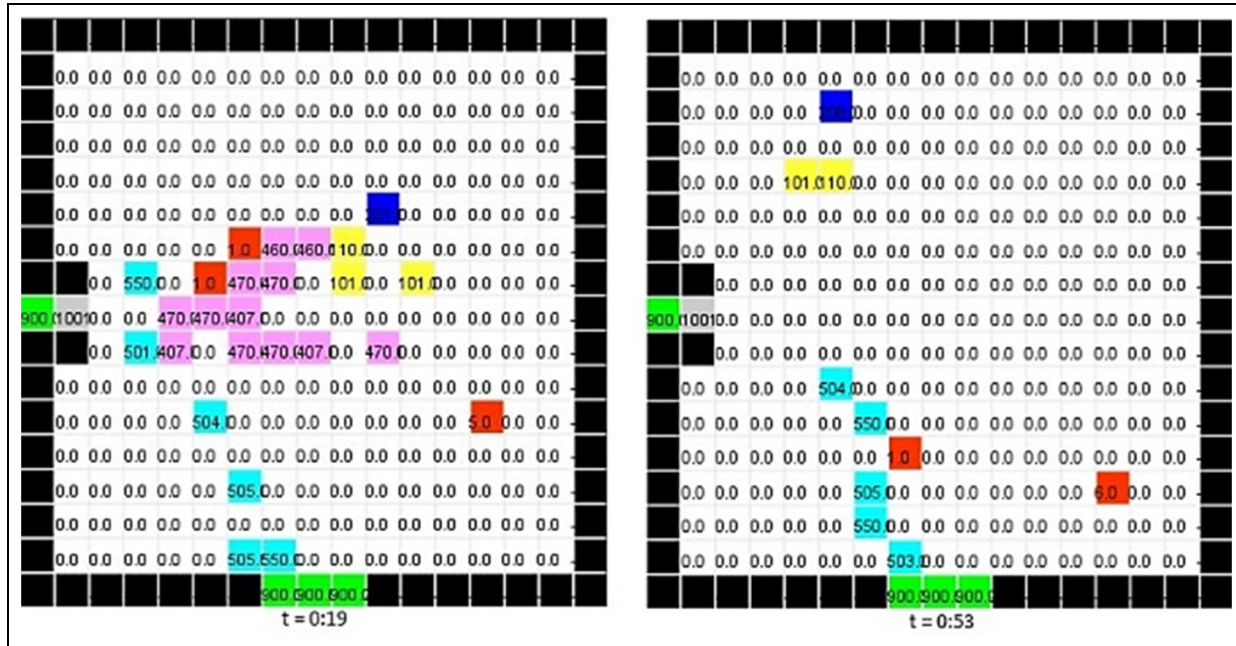


Figure 15. Blocked Exit visualization.

who became stuck in the corner. Future work would be for a more sophisticated panic model to retain information about the length an occupant has been in a panicked state, and perhaps introduce rules to govern how they return to a more focused state. Figure 15 shows snapshots of the Blocked Exit Model completing its simulation. It shows how the herd response can still be slow enough for occupants to remain stationary and enter a panicked state.

**6.3.3 Crowd Control.** The Crowd Control Model source file was 566 LOCs on completion, with 353 of those lines represented as Cell-DEVS rules. This represents the final iteration of the model suite, a single model that has integrated nearly every other model in the suite. Table 14 records the results of the simulation execution for the Crowd Control Model.

### 6.4 Airport Egress results

The Airport Egress Model suite is a collection of source files, almost one for each model previously assessed, where the Daytona Beach Airport provides the foundation for the cell neighborhood. Each model source file represents a single model stitched together from four separate sub models that represent four areas of the Daytona Beach Airport: Check-In, Lobby, Baggage, and Plaza. Figure 9 shows the cellular representations of these areas.

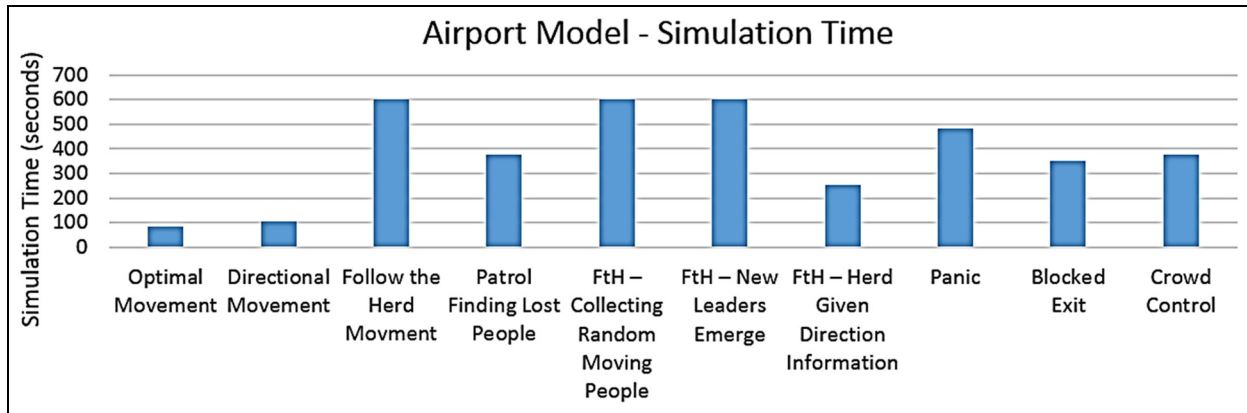
The purpose of this section is to provide a comparative assessment of the models developed in the previous three sections. While the cellular structure had to change for

different models to demonstrate various model functionalities, the Airport Egress Models use the same structure, with the same number of occupants in the same positions for each model. There will be discrepancies with regards to the determination of occupant types in the case where models have multiple occupant types. Also, a decision was made to not test the Random Movement Model, given the size of the cellular neighborhood, and to not test the Patrol Movement Model, as this model is strictly a functionality demonstration and does not provide any value unless the patrol is actively searching for people. Table 15 records the results of the simulation time for each model to fully evacuate the airport.

Finally, looking at Table 15, we can begin the process of a comparative assessment between the models. The development of an environment approximating an airport allows for the modeling of situations that occupants may actually face compared to the ideal conditions provided in the first three sections. It is still not a true comparative assessment given that these models have not been validated; however, the work is not nugatory as it does provide some insight into the proportional responses for each model. Figure 16 provides a graphical representation of this comparative assessment.

Given the time it was taking to compile each model, a decision was made to put a cap of 10 simulated minutes on the total egress time. The initial FtH Movement Model, the FtH – Collecting Random Movers Model, and the FtH – New Leaders Emerge Model all exceeded the 10 minute cap. This was to be expected, as these three models are not given any information with regards to the direction of the





**Figure 16.** Comparison of Airport Egress Models. FtH: Follow the Herd.

**Table 14.** Crowd Control simulation.

Simulation time	Computation time	Memory usage
1:04 minutes	22.54 seconds	28 MB

exits, and occupants are easily led into corners and other tight spots far from the exit.

The Optimal Movement and Directional Movement Models lead the pack, which is to be expected given that all occupants in these models have access to knowledge directing them to the exits. The FtH Herd Given Direction Model was the next best performing model, which makes sense given that it is the first integrated model to give exit information to the occupants while not being restricted by patrol routes. The Blocked Exit Model provided the next best performance. It was expected to take longer than the FtH - Herd Given Direction Model, as the use of an exit has been removed; however, a larger delay was expected. The smaller delay than what was expected may simply be due to the structure of the airport.

The Patrol Finding Lost People and Crowd Control Models finished at the same time. This was also to be expected as the patrol routes used for these two models were the same, and all other occupants reached the exit before the patrolling occupants finished their patrol routes. Finally, other than the models that went over the limit, the Panic Model had the worst response. This was expected as it was a complication on top of the FtH Herd Given Direction Model, and it did not have the authority figures to provide directional guidance like in the Blocked Exit and Crowd Control Models.

## 7. Conclusions

In this research project the concept of modeling and simulation of human behavior has been explored using an

efficient CA approach, the Cell-DEVS formalism, to develop a suite of egress models. This project addresses the existing literature on human behavior to establish a set of movements and behaviors that could then be translated into models and simulations using the Cell-DEVS formalism. The CD++ toolkit provided a suite of tools allowing for the translation of Cell-DEVS models into source code that could then be compiled, simulated, and visualized in order to help build an intuitive understanding of the mechanics that govern evacuation and egress during an emergency situation. Finally, this suite of models has been packaged for upload to the DEVS community portal where they will be freely and publically available for those who wish to utilize or further improve these models.

The number of movements and human behaviors driving the development of models in this research project is not exhaustive. The goal of this research project was to provide something foundational that could then be built upon, so there exists scope for researchers to take the work completed in this project and expand upon it.

The models in this suite were built on the assumption that occupants were evacuating public or commercial areas, such as malls or airports. The mechanics that drive evacuation in other areas, such as personal dwellings, would require the modeling of different behaviors, such as scavenging for important keepsakes and personal belongings. This expansion of the model suite into personal dwellings would be applicable and would provide value to emergency planning for areas with multiple dwellings, such as apartment complexes.

Finally, one of the key components not addressed in this research is validation of this suite of models. As opposed to validation of models representing hardware or software, where the task can be conceptualized and broken down into manageable activities, the validation of models approximating human behavior is a complex undertaking that requires a range of humans as resources. Humans do not always provide consistent inputs and outputs, hence the difficulty in

**Table 15.** Airport Egress Model simulation.

Model type	Simulation time (min)	Simulation time (s)
Optimal movement	1:24	84
Directional movement	1:44	104
Follow the Herd movement	Greater than 10	600
Patrol Finding Lost People	6:18	378
FtH – Collecting Random Moving People	Greater than 10	600
FtH – New Leaders Emerge	Greater than 10	600
FtH – Herd Given Direction Information	4:14	254
Panic	8:01	481
Blocked Exit	5:50	350
Crowd Control	6:18	378

FtH: Follow the Herd.

developing models that approximate human behavior, but with enough time and resources, validation of models approximating human behavior can be achieved. Validating this suite of models would represent a significant step towards the application of this suite of models in the aid of building egress design and evacuation planning for emergency control authorities. One of our future works is to consult with Public Safety officers at Daytona Beach International Airport to conduct a series of validation activities. We envision their input to further signify our results and tune our models' parameters accordingly.

All the models generated during this research are publicly available at Egress Cell-DEVS Models Source Code.<sup>4</sup> Videos demonstrating simulation runs are also available at Egress Simulation Videos.<sup>19</sup>

### Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

### References

- Pan X. *Computational modeling of human and social behaviors for emergency egress analysis*. Doctoral Dissertation, Stanford University, 2006.
- Zeigler BP, Praehofer H and Kim TG. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. San Diego, CA: Academic Press, 2000.
- Wainer GA. *Discrete-event modeling and simulation: a practitioner's approach*. Boca Raton, FL: CRC Press, 2009.
- Egress Cell-DEVS Models Source Code. [https://github.com/sjafer/Egress\\_Cell-DEVS\\_Models.git](https://github.com/sjafer/Egress_Cell-DEVS_Models.git) (accessed 28 June 2016).
- Lawler R and Jafer S. Egress modeling with cellular discrete event system. In: *SoutheastCon 2015*, IEEE, April 2015, pp.1-8.
- Tsai J, Fridman N, Bowring E, et al. ESCAPES: evacuation simulation with children, authorities, parents, emotions, and social comparison. In: *The 10th international conference on autonomous agents and multiagent systems-volume 2*, Taipei, Taiwan, May 2011, pp.457-464. International Foundation for Autonomous Agents and Multiagent Systems.
- Radianti J, Granmo OC, Bouhmala N, et al. Crowd models for emergency evacuation: a review targeting human-centered sensing. In: *2013 46th Hawaii international conference on system sciences (HICSS)*, Maui, HI, January 2013, pp.156-165. IEEE.
- Kuligowski ED, Peacock RD and Hoskins BL. *A review of building evacuation models*. Gaithersburg, MD: US Department of Commerce, National Institute of Standards and Technology, 2005.
- Codd EF. *Cellular automata*. New York: Academic Press, 2014.
- Ma J, Lo SM and Song WG. Cellular automaton modeling approach for optimum ultra high-rise building evacuation design. *Fire Saf J* 2012; 54: 57-66.
- Li D and Han B. Behavioral effect on pedestrian evacuation simulation using cellular automata. *Safety Sci* 2015; 80: 41-55.
- von Schantz A and Ehtamo H. Spatial game in cellular automaton evacuation model. *Phys Rev E* 2015; 92: 052805.
- Ha S, Ku NK, Roh MI, et al. Cell-based evacuation simulation considering human behavior in a passenger ship. *Ocean Eng* 2012; 53: 138-152.
- Peacock RD, Kuligowski ED and Averill JD. *Pedestrian and evacuation dynamics*. New York: Springer Science & Business Media, 2011.
- Song X, Ma L, Ma Y, et al. Selfishness-and selflessness-based models of pedestrian room evacuation. *Phys Stat Mech Appl* 2016; 447: 455-466.
- Wainer G. CD++ : a toolkit to develop DEVS models. *Software Pract Ex* 2002; 32: 1261-1306.
- Farrell R, Moallemi M, Wang S, et al. Modeling and simulation of crowd using cellular discrete event systems theory. In: *Proceedings of the 2013 ACM SIGSIM conference on principles of advanced discrete simulation*, Montreal, Canada, May 2013, pp.159-168. ACM.
- Wang S, Van Schyndel M, Wainer G, et al. DEVS-based building information modeling and simulation for emergency evacuation. In: *Proceedings of the 2012 winter simulation conference (WSC)*, Berlin, Germany, December 2012, pp.1-12. IEEE.
- Egress Simulation Videos. <https://www.youtube.com/channel/UC3TjIsu4lahPqDiVAQBImmA/videos> (accessed 28 June 2016).

**Author biographies**

**Shafagh Jafer** is an assistant professor of Software Engineering at Embry-Riddle Aeronautical University, Daytona Beach, Florida. Dr. Jafer's research focuses on applied modeling and simulation and formal software modeling in the areas of emergency evacuation, natural hazards, and aviation.

**Ryan Lawler**, received his master of software engineering from Embry-Riddle Aeronautical University, Daytona Beach, FL in 2014. He is currently working as an avionics and software engineer at the Australian Defence Force.