Simulation of Aircraft Boarding Strategies with Discrete-Event Cellular DEVS

Shafagh Jafer **Wei Mi**
Department of Electrical, Computer, Software, and Systems Engineering
Embry-Riddle Aeronautical University, Daytona Beach, FL
jafers@erau.edu **miw@my.erau.edu**

Keywords: aircraft boarding, discrete-event, Cell-DEVS, enplane, passenger delay.

Abstract
Time is a critical factor in airlines industry. Among all factors contributing to an aircraft turnaround time, passenger boarding delays is the most challenging one. Airlines do not have control over the behavior of passengers, thus their only key in reducing passenger boarding time is in implementing efficient boarding strategies. In this work we attempt to use discrete-event cellular DEVS to provide a comprehensive evaluation of aircraft boarding strategies. We have developed a simulation benchmark consisting of various boarding strategies including Back-to-Front, Window Middle Aisle, and Reverse Pyramid. Our simulation models are highly precise and adaptive, providing a powerful analysis apparatus for investigating any existing or yet to be discovered boarding strategy. We explain the details of our models and present the results both visually and numerically to evaluate the three implemented boarding strategies. This research is an on-going effort aiming to optimize and reduce passenger boarding delays in commercial aircrafts. .

1. INTRODUCTION

Airlines generate revenue by utilizing and flying airplanes. One of the factors for reducing airlines cost is the quick turnaround of their airplanes. A turnaround time is used to measure the efficiency of airline's operation in a traditional metric. Usually turnaround time is measured by the time between an airplane's arrival and its departure [1]. Some factors that influent the turnaround time include passenger deplaning, baggage unloading, fueling, cargo, airplane maintenance, cargo loading, baggage loading, and passenger boarding. The most difficult factor to control is passenger boarding time since airlines have little control over passengers. Therefore, airlines have to be cautious in making changes to decrease boarding time [2]. Many researchers have proposed and investigated different boarding strategies aiming at reducing the boarding time even by a couple of minutes [2]-[10]. Among the existing boarding strategies the following are the most well-known techniques: Back-to-Front, Random Strategy, Outside-in Method (or Window Middle Aisle), Rotating zone, Reverse Pyramid, Optimal Method, Practical Optimal Method, and Efficient Strategy.

Aiming at evaluating the efficiency of various existing and new boarding strategies, we have implemented a discrete-event simulation benchmark based on the Discrete-Event System Specification (DEVS [12] and Cell-DEVS [14]) formalism. Our benchmark consists of DEVS-based models that are perfectly suitable for executing various simulations on any type of aircraft. Our simulation results are visually presented as 2-D animations, making it easily understandable by non-experts. In this work we only simulate three of the above mentioned boarding strategies: Back-to-Front, Window Middle Aisle, and Reverse Pyramid. We also compare our simulation results to the Optimal strategy to present the degree of efficiency of the given strategies.

This work attempts to present the power of DEVS and Cell-DEVS in analyzing and investigating aviation-related challenges. We demonstrate how flexible, adaptive, and precise Cell-DEVS is in simulating passenger behavior. This paper is organized as follows: Section 2 provides a brief background about various boarding strategies and highlights some of the related works. Section 3 presents our model assumptions. Section 4 discusses the high-level and low-level design. The DEVS and Cell-DEVS implementation details are introduced in Section 5. Section 6 provides the simulation results. The concluding remarks are given in Section 7.

2. BACKGROUND

Since there is yet no "best" boarding strategy, airlines around the world try different methods from time to time. Below we summarize some of the currently available techniques.

Back-to-Front: This boarding plan is known as the "traditional" boarding method. Passengers are boarded to from the back row of the aircraft and continue with the rows up to the front. The zones can be any number reaching from two to the number of actual rows. This strategy is easy to implement, however, it is very likely that it is an inefficient method because congestion is created in a reduced area. [1]

Random Strategy: This boarding plan is when passengers are not assigned to specific seats but line up at the gate counter and are admitted in the order that they arrive. People can choose any unoccupied seat as soon as they get onboard. Passenger will start to rush into plane to get a better seat.

This makes the boarding process faster; however, this reduces the passenger comfort level [4].

Outside- in Method: This method is also called "Window Middle Aisle". Passengers who are assigned to window seats will board first. When it is finished, middle and aisle seats follow. This method has so far revealed very efficient boarding time. It potentially reduces passenger interference caused by loading luggage and completely reduces passengers interfering with each other among the rows. This method is relatively easy to implement [9].

Rotating Zone: This method starts with the last zone in the back to be seated, then continues with the first row in the front. After this, the order continues again with the furthest yet unoccupied zone in the back, then the front one and so on. The advantage of this method is that passengers who are boarding at the back and in the front will not interfere with each other [9].

Reverse Pyramid: This method is to make passenger order from the outer back till the inner front of cabin. This method is in fact a combination of Back-to-Front and Window to Aisle. This strategy is proved to be an efficient method by American West Airlines [2].

Optimal Method: This method is to make passengers board in order from Back-to-Front but in every other row. This methods aims at reducing the interference among passengers from the back and the front, and giving passengers enough space to load their luggage, which reduces the luggage delay in return. However, this method is not practical based on South West Airline experience. It is a challenge to arrange all the passengers in the proper order [8].

Practical Optimal Method: This technique defines four boarding groups. First group is all passengers in even rows in one side of the airplane. The second group is all passengers in even rows in another side of the airplane. The third and fourth group is the passengers in odd rows in each side of the airplane. This method is not as efficient as the optimal method, but it is practical and it proved to be a successfully boarding method [8].

A number of studies have been conducted previously by implementing various boarding strategies using different simulation techniques including: Linear Programming [10], Discrete-Event simulation [2], and even Cellular Automata . A comprehensive literature survey about passenger boarding simulation techniques are reported in [11]. In this work, we have implemented precise and aircraft-independent boarding strategies and provide a comparison of their efficiency. Our simulation is based on the discrete-event DEVS and Cell-DEVS theory. Unlike CA, Cell-DEVS does not require updating the entire cellular grid at every time step. Rather, only cells with updated neighbor values are evaluated. This improvement overcomes the issue of the original CA by reducing the overall execution cost, leading to faster computations. We show that precise simulations results, comparable to those produced by complex mathematical modeling techniques (like those reported in [10] using linear programming), can be obtained from the collective behavior of discrete-event cellular grids. The Cell-DEVS cell space is composed of very simple cells that make local decisions solely based on the information gathered from their immediate neighbors. To implement our DEVS and Cell-DEVS models we used CD++ [13] development toolkit. CD++ is an open-source object-oriented modeling and simulation environment that implements both DEVS and Cell-DEVS theories in C++. The tool provides a specification language that defines the model's coupling, the initial values, the external events, and the local transition rules for Cell-DEVS models. CD++ also includes an interpreter for Cell-DEVS models. The language is based on the formal specifications of Cell-DEVS. The model specification includes the definition of the size and dimension of the cell space, the shape of the neighborhood and the border.

3. MODELING ASSUMPTIONS

Here we present the common parameters and assumptions that were considered for the three boarding strategies (i.e. Back-to-Front, Window Middle Aisle (WMA), Reverse Pyramid) we have implemented. Based on [9], we define ranges for four different parameters as given in Table 1. The first two parameters define the walking speed of a passenger and the time that one passenger needs to sit down at their assigned seat and have their luggage stored in the overhead compartment or underneath the seat in the front. The third parameter is the amount of time a passenger takes to get up of their seat, allowing other passengers to sit within that row. The Passenger flow rate defines the number of passengers that enter the airplane in a certain amount of time. All of these parameters are given in the form of a range from *min_value* to *max_value*. We have mapped these ranges to our Cell-DEVS model to precisely implement a near-reality model of passenger boarding strategies. Details of these mappings are given in the next section when the model's rules are explained.

Table 1. Basic Parameters Ranges. [9]

Parameter	Range	Unit
Walking Speed	0.27…0.44	[m/s]
Clearing Time	6…30	[s]
Get up out of seat	3…4.2	[s]
Passenger flow rate	0.2 …1	[pax/s]

There are two basic elements that interfere the boarding process: aisle interference, and seat interference [9]. Aisle interference is introduced when a passenger is blocked by

another passenger in the aisle, while seat interference is when a passenger tries to get to a seat near the window but is blocked by another passenger already seated near the aisle. Given these two interferences, three types of delays are recognized: walking delay, luggage delay, variable seated passenger delay. For simplicity, at this stage we have implemented our three boarding strategy models with fixed delay times. The corresponding delay values used in our models are given in Table 2.

Table 2. Various Delay Values [9].

Parameter	Time	Unit
Walking delay	2270	ms
Luggage Delay	18000	ms
Two passengers get out of seats	4200	ms
Middle passenger gets out of seat	3600	ms
Aisle passenger gets out of seat	3000	ms

4. SYSTEM DESIGN

Our aircraft model defines an Airbus 320 internal space layout with two different seating areas (first and economy classes) giving a total of 26 rows and 150 seats, a single middle aisle, and an entrance door as given in Figure 1 .

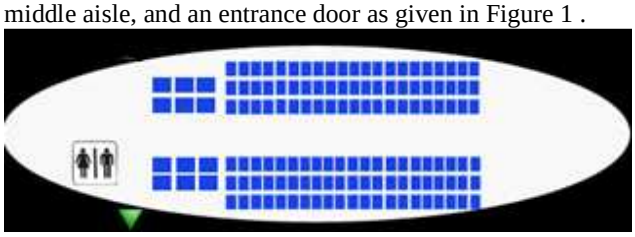

Figure 1. System Overview.

To model the proposed system using DEVS, we define a DEVS coupled model composed of an atomic DEVS component and a coupled Cell-DEVS grid as illustrated in Figure 2. Our DEVS component "*Passenger Generator*" is in charge of generating passengers with specific seat numbers and injecting them into the *Aircraft* cellular model.

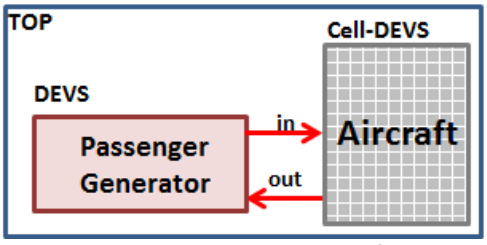

Figure 2. DEVS Conceptual Model of the System.

The two subsystems (Passenger Generator and Aircraft) are interconnected through input/output ports defined in the Top coupled DEVS specification. The atomic Passenger Generator component is defined as follows:

PassengerGenerator= <X, Y, S, δ_{int}, δ_{ext}, λ, ta>
X = {(in, 1)}
Y = {(out, seatNo)}
S = {*waiting, generating*}
δ_{int} (S -> S'){
  if (S = *generating*) -> S = *waiting*
  }
λ (S -> Y) {
  if (S = *generating*) → *out(out, seatNo)*
  }
δ_{ext} (S,e,X -> S'){
  if (S == *waiting* && *e* = *anytime* && X == (in, 1)) → S = *generating*
  }

ta (S -> R^+) { *waiting* → ∞, *generating* → *0*}

Figure 3. Passenger Generator DEVS Specification.

This internal behavior of the generator is translated from its state diagram illustrated in Figure 4. Basically, the Passenger Generator is always in *waiting* mode. Whenever the cellular Aircraft model requests a new passenger entrance (indicated when the aircraft door is not occupied by a passenger), the Passenger Generator calculates an unassigned seat and sends the seat number as a value to the "in" port of the Aircraft Cell-DEVS model. When a seat number arrives at the input port of the door cell, it is simply regarded as a passenger who is assigned that specific seat number.

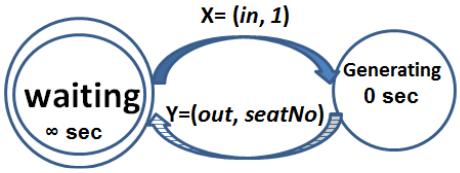

Figure 4. Passenger Generator State Diagram.

Obviously, the seat numbers generated by the Passenger Generator model cannot be duplicates and the order they are sent out to the Aircraft differs from one strategy to another. This is discussed in Section 5.1 – 5.3.

The Aircraft Cell-DEVS model is defined as a coupled DEVS model with 430 cells, where each cell is a DEVS machine. The model's layout is illustrated in Figure 5 where yellow cells denote seats, red cell is a passenger, and gray cells in the middle demonstrate the aisle. The aircraft door is a gray cell in the bottom left corner of the model.

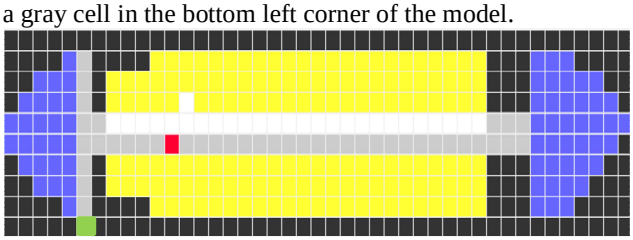

Figure 5. Aircraft Cell-DEVS Model.

The Aircraft model is defined as following:

```
[aircraft]
 type : cell
 width : 10
 height : 43
 neighbors : aircraft(-1,-1) aircraft(-1,0)
 neighbors : aircraft(-1,1)  aircraft(0,-4)
 neighbors : aircraft(0,-3)  aircraft(0,-2)
 neighbors : aircraft(0,-1)  aircraft(0,0)
 neighbors : aircraft(0,1)   aircraft(0,2)
 neighbors : aircraft(0,3)   aircraft(1,-1)
 neighbors : aircraft(1,0)   aircraft(1,1)
```

This yields a 10 by 43 cellular space where each cell defines fourteen cells in its neighborhood, as shown in Figure 6. The cellular neighborhood indicates that the value of a cell is affected by those residing in its neighborhood. Thanks to Cell-DEVS theory, when the value of a cell changes, only its neighborhood cells are notified rather than the entire cell space.

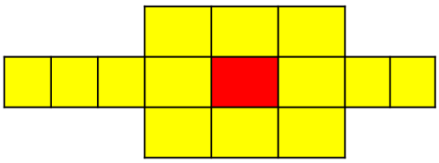

Figure 6. Aircraft Model Cellular Neighborhood.

In order to recognize different cells (passenger, door, aisle, occupied and empty seats, etc.) we have defined our model states as follows (refer to Table 3):

Table 3. Cellular State Values.

State Name	State Value	Color	Description
Wall	0	Black	Wall or obstacle
Aisle (Seat Row Represent)	1 , 51-76	Grey	Passenger aisle
Door Open	2	Green	Boarding door is open
Cabin	3	Blue	Cabin or bathroom or cafe
Empty Passenger Seat	100-3000	white	Passenger seat
Walking Passenger	10,000-300,000	Red	Walking Passenger
Seats with passenger	4	Yellow	Seat is occupied with passenger
Door Closed	9	Green	All the passengers have been boarded.

Figure 7 Demonstrates a screenshot of the aircraft's front where all seats are occupied (yellow), the door is closed (green), and no passenger is in the aisle (grey).

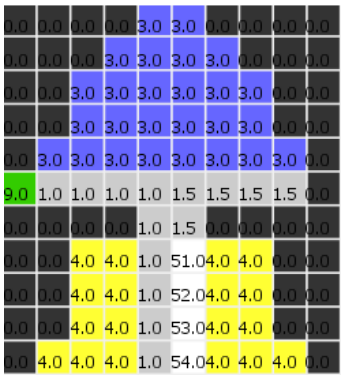

Figure 7. Mapping State Values to Cells.

5. IMPLEMENTATION DETAILS

Seat numbers are generated randomly but are injected into the Aircraft model in a different sequence order depending on the boarding strategy. Thus three different versions of the Passenger Generator model were implemented to accommodate these restrictions. Although, the three models define the exact same DEVS specification that was illustrated in Figure 3, the internal behavior given by the external transition function is slightly different. Next we will present these variations.

5.1. Seat Generation in Back-to-Front Strategy

The Back-to-Front strategy includes six boarding zones: *zone_1* (rows 1 to 3, business seats), *zone_2* (rows 22 to 26), *zone_3* (rows 17 to 21), *zone_4* (rows 12 to 16), *zone_5* (rows 7 to 11), and *zone_6* (rows 4 to 6). Seat numbers are generated based on zones with a simple formula that considers the row number and the seat capacity within that row (6 for economy rows 4-26, and 4 for first class rows 1-3). The code snippet in Figure 8 illustrates how seats within each zone are created upon initialization of the DEVS model.

```
//First class seats
for(int i = 1;i < 4; i++){
  for(int j = 2; j < 6; j++){
    v1.push_back(10000*i+j);}}
```

```
//Economy class: rows 22-26
for(int i = 22; i < 27; i++){
  for(int j = 1; j < 7; j++){
    v2.push_back(10000*i+j);}}
```

```
//Economy class: rows 7-11
for(int i =7; i < 12; i++){
  for(int j = 1; j < 7;j++){
    v5.push_back(10000*i+j);}}
```

```
//Economy class: rows 4-6
for(int i = 4; i < 7; i++){
  for(int j = 1; j < 7; j++){
    v6.push_back(10000*i+j);}}
```

Figure 8. Back-to-Front Seat Number Generation.

Given all seat numbers, upon each request from the Aircraft cellular model, the Passenger Generator model injects a random seat number from within the current boarding zone. In our implementation we simply handle this by shuffling the seat numbers within each zone as following:

```
random_shuffle(v1.begin(), v1.end());
```

With Back-to-Front strategy, random seat numbers are sent out to the Aircraft in the order of *zone_1*, *zone_6*, *zone_5*,

zone_4, *zone_3*, *zone_2*. Only when all seat numbers from a given zone are sent out, the seat numbers from next zone are selected. This behavior is implemented within the DEVS external transition function which is triggered when the Aircraft requests a passenger by sending an input through port "in" of the Passenger Generator model. This is shown in Figure 9.

```
if (!v1.empty()){
   seatNum = v1[0];
   v1.erase(v1.begin());
 }
 else if (!v2.empty()){
   seatNum = v2[0];
   v2.erase(v2.begin());
 }
```

Figure 9. Back-to-Front Random Seats Selection.

5.2. Seat Generation in Window Middle Aisle (WMA) Strategy

The WMA strategy defines four zones: *zone_1* (rows 1 to 3, business seats), *zone_2* (window seats of rows 4 to 26), *zone_3* (middle seats of rows 4 to 26), and *zone_4* (aisle seats of rows 4 to 26). The first zone seats are generated similar to Back-to-Front Strategy, then the seats for the remaining three zones are generated as presented in Figure 10.

```
//First class seats
for(int i = 1;i < 4; i++){
 for(int j = 2; j < 6; j++){
 v1.push_back(10000*i+j);}}
```

```
//Economy class: window
 for(int i = 4; i < 27; i++){
  v2.push_back(10000*i+1);
  v2.push_back(10000*i+6);}
```

```
//Economy class: middle
for(int i =4; i < 27; i++){
  v3.push_back(10000*i+2);
  v3.push_back(10000*i+5);}
```

```
//Economy class: aisle
for(int i = 4; i < 27; i++){
v4.push_back(10000*i+3);
v4.push_back(10000*i+4);}
```

Figure 10. WMA Seat Number Generation.

As discussed in Back-to-Front strategy, the seat numbers are sent to the Aircraft by selecting random seat numbers from within each zone, given a zone sequence of: first class (*zone_1*), window seats (*zone_2*), middle seats (*zone_3*), and aisle seats (*zone_4*). Only when a zone is completely seated, the next zone is selected for seating (random fashion is only within each zone, the zones follow WMA sequence).

5.3. Seat Generation in Reverse Pyramid (RP) Strategy

Similar to Back-to-Front, the RP strategy defines six zones: *zone_1* (rows 1 to 3, business seats), *zone_2* (window seats of rows 13 to 26), *zone_3* (window seats of rows 8 to 12 and middle seats of rows 18 to 26), *zone_4* (middle seats of rows 8 to 17 and window seats of rows 4 to 7), *zone_5* (aisle seats of rows 17 to 26 and middle seats of rows 4 to 7), and *zone_6* (aisle seats of rows 4 to 16). Figure 11 provides the implementation of *zone_3* and *zone_4*.

```
//zone3
for(int i = 8; i < 13; i++){
 v3.push_back(10000*i+1);
 v3.push_back(10000*i+6);
   }
for (int i = 18; i < 27; i++){
 v3.push_back(10000*i+2);
 v3.push_back(10000*i+5);
   }
```

```
//zone4
for(int i = 8; i < 18; i++){
 v3.push_back(10000*i+2);
 v3.push_back(10000*i+5);
   }
for (int i = 4; i < 8; i++){
 v3.push_back(10000*i+1);
 v3.push_back(10000*i+6);
   }
```

Figure 11. RP Seat Number Generation.

Unlike the DEVS Passenger Generator model which has to behave differently under various boarding strategies, the Aircraft Cell-DEVS model is exactly the same. The following section reveals its details.

5.4. Aircraft Rules Specification and Implementation

Based on the Cell-DEVS model defined in Section 3, the Aircraft model implements a series of rules that are evaluated for every cell on the cell space over time steps. As mentioned before, the three boarding strategies use the exact same Cell-DEVS Aircraft model, since the boarding pattern really depends on the order at which passengers are called to enplane. This is handled by the DEVS Passenger Generator model described in the previous section. Here we will present the cellular model rules and explain how simulation evolves based on the discrete-event and continuous-time property of DEVS theory.

The rules are divided into two groups:

1) *pre-seat rules*: a set of 9 rules with responsibilities to send requests to the Passenger Generator model to release passengers, and guide passengers at the aircraft door to walk to the beginning of the seats aisle. The area that *pre-seat* rules applies to is from cell (0, 0) to cell (6, 0) where cell(y, x) defines the *y* and *x* coordinates of the cell on the grid. The affected area by *pre-seat* rules is highlighted in a surrounding solid box in Figure 12.
2) *seating rules*: a set of 33 rules handling passengers' forward movement within the aisle and occupation of seats. These rules only apply to the cells that represent the seats (both first class and economy) and the aisle, as well as passengers on these cells. This area is highlighted in Figure 12 with a surrounding dashed box.

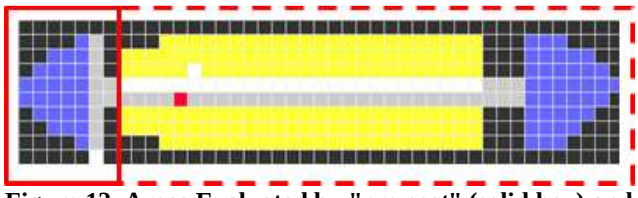

Figure 12. Areas Evaluated by "*pre-seat*" (solid box) and "*seating*" (dashed box) Rules.

A rule in Cell-DEVS is the local computing function which is defined in the form of {*result*} *delay* {*precondition*}. This

indicates that when the *precondition* is met, the state of the cell changes to the designated *result* after the duration specified by *delay*. If the precondition is not met, then the next rule is evaluated until a rule is satisfied or there are no more rules. In the space below we will present some of the rules implemented in Aircraft model.

For instance, the following rule (from *pre-seat* rules):

```
rule: {(0,0)+send(out1,2)} 0 {(0,0)=2}
```

defines that whenever the door cell is unoccupied, a request for passenger entrance should be sent to the Passenger Generator model immediately.

Now let's consider the *seating* rules for a scenario where a passenger is walking down the aisle with a window seat assigned to her. There are four possible scenarios, thus four evaluation rules:

1) Aisle seat and middle seat is occupied:

```
rule: 1 #Macro(WBothSeatDelayAddUp)
   {(0,0)> 10000 and (0,-1)=4 and (0,-2)=4}
```

2) Only aisle seat is occupied

```
rule: 1 #Macro(WAisleSeatDelayAddUp)
{(0,0)>10000 and (0,-1)=4 and (0,-2)>100}
```

3) Only middle seat is occupied

```
rule: 1 #Macro(WMiddleSeatDelayAddUp)
{(0,0)> 10000 and (0,-2)=4) and (0,-1)>100}
```

4) Neither the aisle seat nor the middle seat is occupied

```
rule: 1 #Macro(WNoneSeatDelayAddUp)
{(0,0)>10000 and (0,-1)>100 and (0,-2)>100)}
```

The #Macros defined in the above rules are the fixed delays applied to the passenger when he/she gets to the assigned row. These delays are defined in a "boarding.inc" file with a format presented in . The delay values are conducted from the literature and are addressed in Table 2.

```
Time for putting off the carry on luggage
#BeginMacro(luggageDelay)
18000
#EndMacro

Section for adding up time
Time for window passenger to get in with both
passenger in seats {4200+2270*3+18000}
#BeginMacro(WBothSeatDelayAddUp)
33210
#EndMacro

Time for window passenger to get in with middle
passenger in seats  {3600+2270*3+18000}
#BeginMacro(WMiddleSeatDelayAddUp)
28410
#EndMacro
```

Figure 13. Macros Defining Delay Values.

Due to space limitation, we are not able to show all the rules, however the logic and the format is very similar to what we just presented above.

6. SIMULATION RESULTS

CD++ also provides a visualization tool, called *CD++ Modeler*, which takes the result of the Cell-DEVS simulation as input and generates a 2-D representation of the cell space evolution over the simulation time (presented in Figure 14). We will use this feature to visually present the results of our simulations.

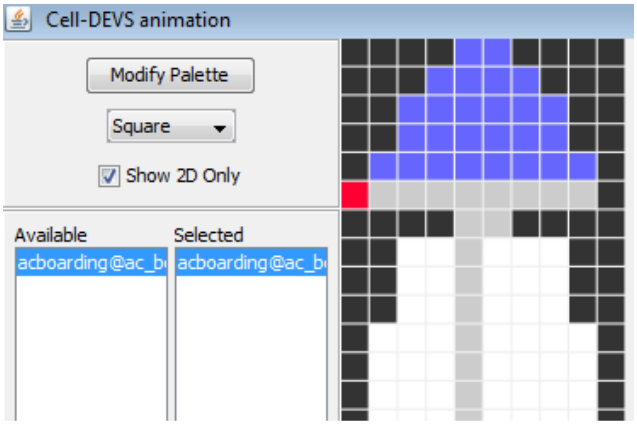

Figure 14. CD++ Animation for Cell-DEVS.

Given the common Cell-DEVS model file ("Aircraft.MA") we execute the overall simulation by including the desired Passenger Generator DEVS model for that specific boarding strategy (Back-to-Front, WMA, and RP). The simulation results are captured in the following screenshots.

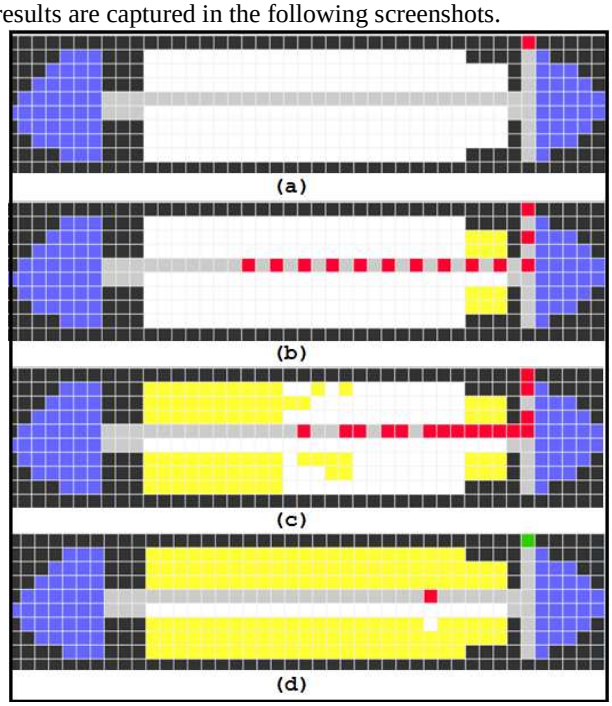

Figure 15. Back-to-Front Strategy.

Figure 15 shows four simulation scenarios of the Back-to-Front strategy: (a) one passenger has entered the aircraft and currently occupying the door, (b) the first class zone is completely seated, (c) the last back two zones are also seated, (d) the last passenger is about to be seated.

Similar simulation results were also conducted for the other two strategies. As illustrated in Figure 16 for the WMA strategy, the four scenarios describe when: (a) the first class passengers are seated, (b) all window passengers are seated, (c) all middle passengers are seated, (d) the last three aisle passengers are about to be seated.

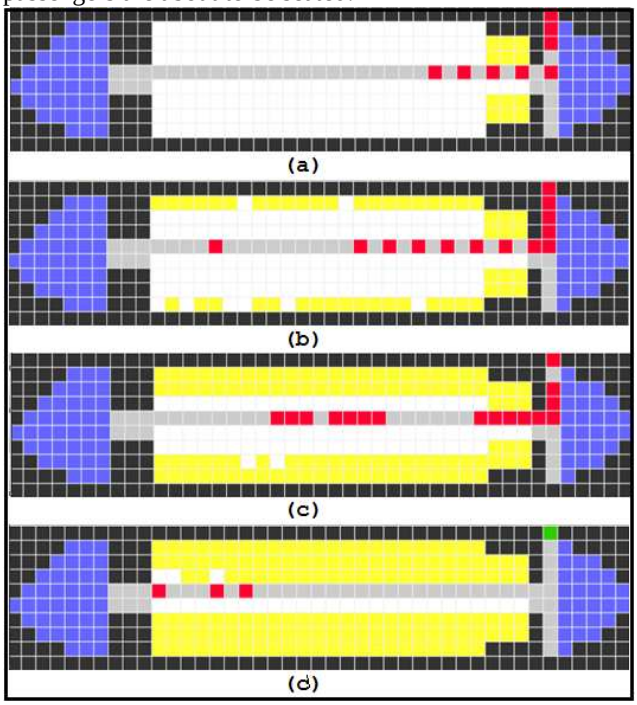

Figure 16. Window Middle Aisle (WMA) Strategy.

Finally, the Reverse Pyramid (RV) simulation screenshots are presented in with (a) to (d) seating sequence.

In order to compare the performance of the implemented strategies, we collected the overall execution time for the simulations. As demonstrated in Figure 18, the Window Middle Aisle strategy is the most efficient seating pattern with an overall simulation time of 26.16 minutes. In order to compare our results to an ideal (but not practical) strategy, we had also implemented an Optimal strategy where passengers were seated in a descending order starting from the back of the aircraft. Under this strategy, each row was seated in the windows-middles-aisles order, one row at a time. Clearly, this strategy is not practical since it will cause huge line ups at the gates, trying to get passengers entering the airplane one by one at a descending order of seat number (with exception for first class passengers).

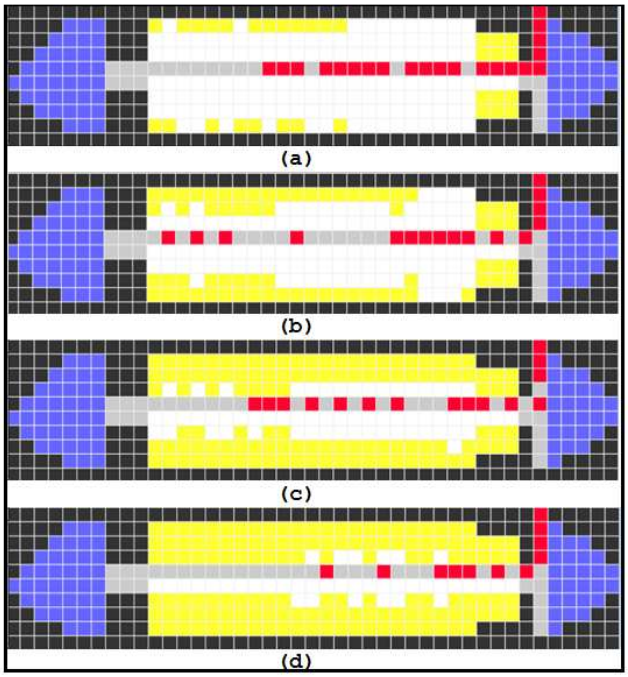

Figure 17. Reverse Pyramid Strategy.

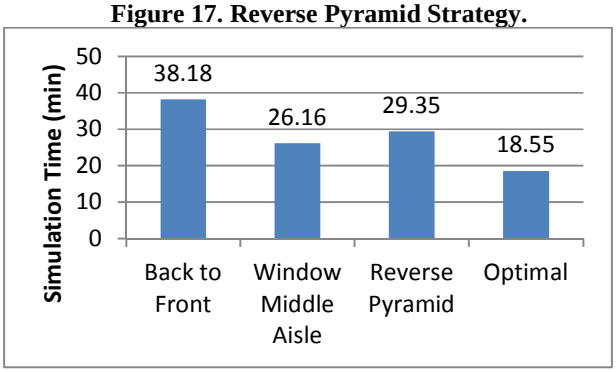

Figure 18. Strategies Comparative Results.

7. CONCLUSION

We presented a simulation study investigating the efficiency of aircraft's passenger boarding strategies. We used the Discrete-Event DEVS and Cell-DEVS theory to evaluate three boarding strategies: Back-to-Front, Window Middle Aisle, and Reverse Pyramid. With the obtained simulation results, we concluded that among the three implemented boarding strategies, the Window Middle Aisle provided the least boarding time, while the Back-to-Front was the most inefficient method. Given the ideal smallest boarding time of 18.55 minutes from the Optimal Strategy, our most efficient boarding strategy exceeds this by about 7 minutes which in airlines world it is still a significant time incurring huge costs. We are currently implementing other boarding strategies, as well as variable walking and luggage storing delays, aiming at exploring better options to save time, satisfying both passengers and airliners.

8. REFERENCE

[1] Briel, M. H., Villalobos, J., & L.Hogg, G. "The Aircraft Boarding Problem". 12th Industrial Engineering Research Conference (2003).

[2] Menkes, H.L., Van den Briel, J. V. "American West Airlines Develops Efficient Boarding Strategies". Interfaces, 35:191 – 201, 2005.

[3] Ferrari, P., Nagel, K. "Robustness of efficient boarding in airplanes". Transportation Research Record, 1915:44–54, 2005.

[4] McFadden, L. C. "A Study of Airline Boarding Problem". Journal of Air Transport Management.2008.

[5] Van Landeghem, H., Beuselinck, A. "Reducing passenger boarding time in airplanes: a simulation based approach". European Journal of Operational Research, 142(2):294–308, 2002.

[6] Marelli, S., Mattocks, G., Merry, R. "The role of computer simulation in reducing airplane turn time". Aero Magazine, 1998.

[7] Nyquist, D.C., McFadden, K.L. "A study of the airline boarding problem. Journal of Air Transport Management". 14:197–204, 2008.

[8] Steffen, J. H. "Optimal boarding method for airline passengers". Journal of Air Transport Management, 14:146 – 150, 2008.

[9] Muller, J. "Optimal Boarding Methods for Airline Passengers". Haumburge University Internal Report. 2009.

[10] Bazargan, M. "A Linear Programmming Approach for Aircraft Boarding Strategy". ScienceDirect. 2007.

[11] Audenaert, J., Verbeeck, K., Berghe, G.V. "Multi-agent based simulation for boarding". Proceedings of the 21st Belgian–Netherlands Conference on Artificial Intelligence. pp. 3–10. 2009.

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

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

[14] G. Wainer, "Discrete-Event Modeling and Simulation: a Practitioner's approach". CRC Press. Taylor and Francis. 2009.