Modeling Hatton Cross Roundabout system with Cell-DEVS

Peter Miebach **Pradeep Gunaratnam**

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6 Canada
{petermiebach, pradeepkumargunaratn}@cmail.carleton.ca

Keywords: Discrete event simulation, DEVS models, Cell-DEVS, Cellular automata, Hatton Roundabout system

Abstract

Hatton Cross Roundabout (HCR) is a modern roundabout where large ring of road composed of six mini roundabouts. In this paper, CD++ Cell-DEVS and Rise software are used for simulation to reproduce the behavior of Hatton Cross Roundabout system, and to further identify the factors that affect the throughput of the roundabout system. Cell-DEVS is an extension of DEVS that supports defining and executing cellular automata models [1][2].

Firstly, this paper will introduce the project, idea and the background. Then, it will provide detailed information about the background needed for the project. Model definition is presented in the next section followed by simulation results. Based on the simulation results, a conclusion will be provided.

1. INTRODUCTION

A type of circular intersection or junction in which road traffic is slowed and flows almost continuously is called a roundabout. Hatton Cross Roundabout (HCR) is a modern roundabout where large ring of road composed of six mini roundabouts as shown in the figure 1 below. Two lanes in each direction to its neighbor link, which is indicated in blue box in the figure 1, connect with the other mini-roundabouts. Therefore, approaching it, you can turn either left or right on to the "roundabout" and proceed around it in whichever direction provides the shortest route to your exit. Traffic flow around the smaller, inner roundabout is clockwise, and traffic flows in the usual counter-clockwise manner around the six mini-roundabouts and the outer loop. The dashed line represents the yield point where entering traffic must always yield to traffic already in the circle regardless of lane position. It is assumed that a car would know its destination exit before it enters the Hatton Cross Roundabout. In order to simulate that, random exit number is assign to each car as their destination exit before allow them into the HCR [3][4].

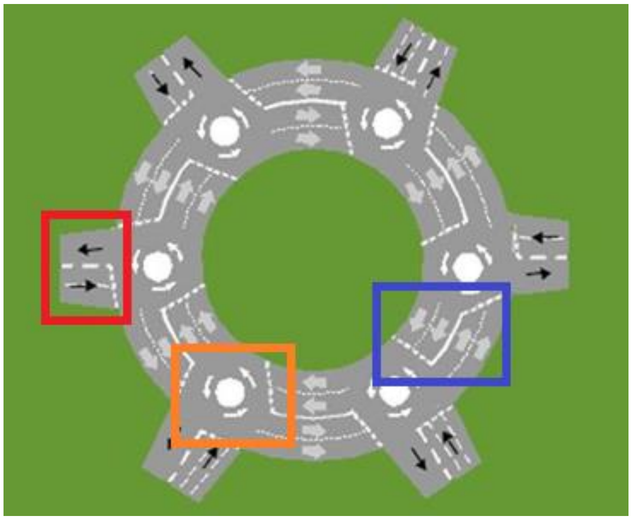


Figure 1: Hatton Cross Roundabout [4]

The red square in the figure 1 shows one of the exits in the one of the mini-roundabouts. In Hatton Cross Roundabout, exiting directly from inner lane is generally permitted. In order to simplify the problem, we made the assumption that we can only take exit from the outer lane. The exiting traffic has the right-of-way over entering traffic. On the other hand, exiting from inner loop directly in the traffic circle, which is shown in the orange square in figure 1, is usually not allowed. One must change lane to the outer loop before exiting.

The purpose of this simulation is to use CD++ Cell-DEVS and RISE software to reproduce the behavior of Hatton Cross Roundabout system, and to further identify the factors that affect the throughput of the roundabout system. For this project, some of the components and models that we implemented in the assignment 2 will be reused as a base and we will enhance it to meet the project expectations.

2. BACKGROUND

The model chosen for assignment 2 is a two-lane circular mini-roundabout traffic intersection, which is similar to the one of the mini circles in the Hatton Cross Roundabout in the figure. The example was extracted from the paper [5]. The figure 2 below shows the mini-roundabout that we implemented in assignment 2. The arrow shows the direction of the traffic flow in the round about.

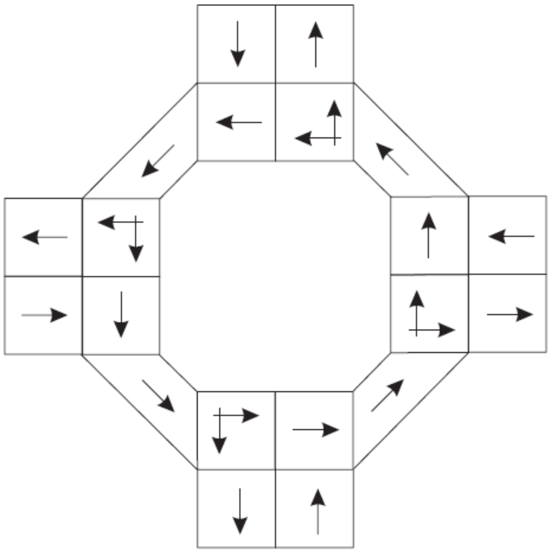
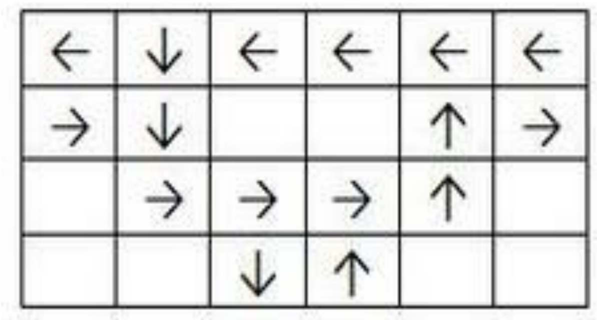


Figure 2: Traffic Direction within a Two-Lane Roundabout (assignment 2) [5]

We will modify this model to fit to our project requirement where only three exits to the mini-roundabout and two exits connect to other mini-roundabouts in each direction. The figure 3 below shows the modified version of the mini-roundabout that will be used in the Hatton Cross Roundabout for this project. The left and the right exits connect with the other mini-roundabouts while the bottom exit connects with the road external the Hatton Round About system. Special rules will be used for connecting the mini roundabouts to decide while lanes the traffic should be fed into the next roundabout. It will consider how many hops the car is away from its destination exit.

A traffic queue model will be joined in the exits in which connect with roads outside of the system in order to form a new Cell-DEVS model. The queue delivers the car to the first entrance cell of the mini roundabout. When it is safe to enter the roundabout, the car will be move from the entrance cell into the mini roundabout circle of cells. Each car enters the roundabout one at a time. A car's velocity is predetermined, it is an indication of the driver's speed at which they prefer to travel at. Before a car that enters the system, it will be given a random exit point at which the car will leave the roundabout, the number of mini roundabout the car must pass through to reach that exit point, and the speed of which the driver likes to drive at. Car entering traffic must always yield to traffic already in the circle, and enter the roundabouts when it is safe, by perceiving the other car's velocity and position. The traffic will flow in the counter-clock wise inside the circle.



←	↓	←	←	←	←
→	↓			↑	→
	→	→	→	↑	
		↓	↑		

Figure 3: Traffic Direction rule for mini-roundabouts in HCR (project)

In Hatton Cross Roundabout model, each mini-roundabout and roundabout connectors will be represented as Cell-DEVS model while the roundabout entrance to the system will be characterized as a queue DEVS models as discussed above[6].

As part of the conceptual definition of the system, we make the following assumptions that can serve as the boundary conditions of the system, which further form the so-called:

1. When a car exiting the roundabout circle to another mini roundabout doesn't need to wait.
2. A car can only travel from one cell to its adjacent cell ahead except at the exit point where it can travel to right/exit cell and exit the system.
3. The car within the mini roundabout will leave at the destination exit that it was assigned.
4. When the car leaves a mini roundabout that is connected to another mini roundabout, it will decrease its 'number of mini roundabout hops left' by 1. If the number of mini roundabout hops is 0 when the car enters the mini roundabout, its destination will be changed to leave the system.
5. The out-going exits will be free always and the car doesn't have to wait to exit out of the system.

6. The capacity of the roundabout area is limited; the maximum number of vehicles it accommodates is constant and defined when the system is built.
7. No vehicle stops in the roundabout circle area at any given time unless there is no free adjacent cell ahead.
8. Each vehicle in traffic uses the same amount of time to pass through the intersection area.
9. Except traffic roundabout circle and the yield lines, there are no other traffic control methods considered in the system.
10. The traffic in each direction has at least one lane respectively.
11. From each exit, only one car will enter the HCR system at a given time.
12. The length of each lane, that feeds the traffic into the HCR, is unlimited and each lane is capable of accommodating unlimited number of vehicles at a given time. Therefore, unlimited number of vehicle will approach each exit to enter the system.
13. The entering traffic will be queued in the each exit and will be send into the system one by one as the space available in the system

3. MODEL DEFINITION

The roundabout is composed of three main models: Input Queue mode, mini-roundabout connector model and roundabout models. Each of this models and how they are represented is discussed below. This section also discussed about the formal definition

3.1. Input (Entrance) Queue

This queue is used to store all cars in that enter the specific entrance into the HCR system. It can contain zero or more car at any time. When the cell in front of the exit becomes free the car will be removed from the queue and send into the system. Each car in the queue will be assigned a destination exit number, number of mini roundabouts it must pass through and the driver's level of speed. The entrance part is the one feeds the traffic from outside of the HCR roundabout to the roundabout. It also, sends the traffic outside of the HCR roundabout from the inside of the system. Before it feeds the traffic into the system, it verifies if the system has capacity to take another car and the cell in front is empty. If yes then it will take the car from the input queue and sends it into the system. This will be represented by Queue DEV model.

3.2. Mini-Roundabout connector

The mini-roundabout connector connects the two mini-roundabouts. One of the exits from one mini-roundabout is connected to other mini-roundabouts exits that receive traffic input. The figure 1 shows how it is connected. To which lane of the exit the car should be fed decided based on how many hops the car is away from its destination exit. This will be represented by Cell-DEVS model

3.3. Roundabout

This component represents one of the mini-roundabouts where a circle road exists. The lane and the area of roundabout, through which the traffic of that lane passes, are stateless. A car can move around the roundabout if no car is ahead. Cars move in a counterclockwise direction. Cars entering the roundabout yield to cars already present in the roundabout that as coming towards those cars. The mini roundabout component will also be represented by Cell-DEVS model.

3.4. Formal Definition

In this section, the specification of the models will be discussed one by one.

The DEVS formal specification, $<X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta>$ for each of the atomic models is as follows:

3.4.1. Traffic Input Queue (atomic)

We will have six instance of Traffic Input Queue. As be explained in the previous section, it will be coupled with the external exit of the HCR. The names of the each instance are: TrafficInput_1, TrafficInput_2, TrafficInput_3, TrafficInput_4, TrafficInput_5 and TrafficInput_6.

The formal definition of the traffic input queue is shown below:

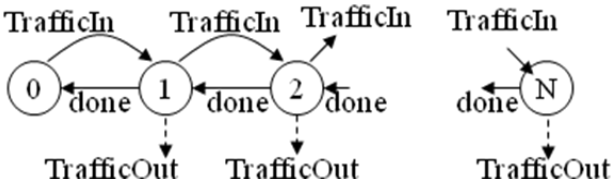


Figure 4: Traffic Input queue

$M = <X, Y, S, \delta, \lambda>$
$X = \{TrafficIn, done\}$
$Y = \{TrafficOut\}$
$done = TrafficOut$
$S = \{queue.length \text{ in } [0,N]\}$
$\delta : X * S \rightarrow S$
$\delta(TrafficIn, n) = n+1 \text{ for all } 0 \leq n < N$
$\delta(done, n) = n-1 \text{ for all } 0 < n \leq N$
$\lambda : S \rightarrow Y$
$\lambda(n \neq 0) = TrafficOut$

The traffic input queue module will be tested using the "black box" testing method. Test cases are created by adding inputs at different times. The queue boundaries will be tested. It will be verified that the queue state increments as new traffic arrives and decreases as traffic leaves. When there is no traffic in the system it shows appropriate state[7].

3.4.2. Cell-DEVS formal specifications

The formal specification for the Cell-DEVS is discussed in this section.

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

Cell's Neighbours

M = <Xlist, Ylist, I, X, Y, η, N, {r,c}, C, B, Z, select>
Xlist = {(-1,-1)…(1,1)};
Ylist = {$\emptyset$};
$I = < ?, \mu^x, \mu^y, p^x, p^y >$
N = {(-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-2), (1,-1), (1,0), (1,1), (1,2), (2,-2), (2,-1), (2,0), (2,1), (2,2)};
Ƞ = 25;
r = 8; c = 20;
X=Y={1, 2, 3, 4, 5}:
B = not-wrapped
C = {Cij | i ϵ [1, 6], j ϵ [0, 19]};
Z: Inverse neighborhood of N
Select = {(0,1), (0,-1), (-1,0), (1,0)};

Cell-DEVS models also planned to be tested using black box where different inputs will be given to each sub-component. Then, the results will be compared with the expected result to see the component is functioning properly. This will be done in much iteration.

3.5. Model variables

Cell state variables are used to represent the direction of the cell, isExitPort status, initialization, and the last for testing. The first digit is direction (D), the second digit is isExitPort (E) status, third digit is Init (I) and fourth one is Testing (T)

State variables composition: D E I T

The possible values for D, E, I and T are described below. What each of the value indicates is also discussed below.

3.5.1. Cell Direction (D)

This variable describes the direction of the cell. The possible value and what it meant by the value is listed below.

- 1 (North)
- 2 (East)
- 3 (South)
- 4 (West)

3.5.2. isExitPort (D)

This field specifies if the cell is an exit and which one. The possible values for "isExitPort" field and what it means are listed below.

- 7 (Exit West)
- 8 (Exit South)
- 9 (Exit East)
- 0 (Not an Exit)

3.5.3. Initialization (I)

In order for cells to output their characteristics at run time, initialization of directionPort and isExitPort is set by the state variables direction and isExit. Initially "I" is zero and after the first iteration through the cell runs it becomes 1. It remains 1 for the remainder of the simulation.

3.5.4. Testing (T)

This value is used for testing. Used when inserting a car into a cell and T is a flag to insure that the car is only inserted on the first iteration of the rules on the cells. When T is used, the car will be inserted in certain cells with set values such as exit number, number of hops left and speed.

3.6. Initial state variable values

The initial state variable values are shown below. A rule was created to set up the initial state variables.

(1,1) = 4 7 0 0 //cell (1,1) has direction of west and it is exit west
(1,2) = 3 0 0 0 //cell (1,2) has direction of south and it is not an exit
(1,3) = 4 0 0 0 //cell (1,3) has direction of west and it is not an exit
(1,4) = 4 0 0 0 //same way following states can be interpreted
(1,5) = 4 0 0 0 //the last zeros represent init and test respectively
(1,6) = 4 0 0 0
(2,1) = 2 0 0 0
(2,2) = 3 0 0 0
(2,5) = 1 0 0 0
(2,6) = 2 9 0 0
(3,2) = 2 0 0 0
(3,3) = 2 0 0 0
(3,4) = 2 0 0 0
(3,5) = 1 0 0 0

(4,3) = 3 8 0 0
(4,4) = 1 0 0 0

3.6.2. Initial values for exitNumberPort:

The initial row value for the exitNumberPort neighbor port is shown below.

```
initialvalue : 0
localtransition : roundabout-rule
initialrowvalue :  0     00000000
initialrowvalue :  1     01111110
initialrowvalue :  2     01100110
initialrowvalue :  3     00111100
initialrowvalue :  4     00011000
initialrowvalue :  5     00000000
```

Note that 1 represents an empty road and 0 is not a road.

3.7. Cell Input and Output Neighbour ports

The input and output neighbor ports detail is discussed in this section. The car characteristic is influenced by Car Exit Number, Number of Hops, Speed, direction, and isExit neighbor ports. The neighbor ports are as follows: exitNumberPort , numHopsLeftPort, speedPort, directionPort, and isExitPort. Each of these variables and the possible values for them are discussed below.

3.7.1. exitNumberPort

The following five values are used for car exit number. What meant by car exit number is that, which exit the car is going to take. The value zero means it is not a road. This value is given to the car before in enters the system and this values is passed along as car moves until it exits the system.

- 1 (Empty Road)
- 7 (Cell occupied by car going to West Exit)
- 8 (Cell occupied by car going to South Exit)
- 9 (Cell occupied by car going to East Exit)
- 0 (Not a road a.k.a. border)

3.7.2. Number of Hops

The number of hops left neighbor ports indicates how many mini roundabouts car must pass through before it reaches its desired destination exit. In the case for the Hatton Cross roundabout, the possible values for this is: 0, 1, 2, 3, 4 and 5

3.7.3. Car speed

The speed variables show the speed of the car. There are two speed levels available which are low and high. They are represented by 1 and 2 respectively. Typically car will have a value of 1.

When a car is waiting to enter the roundabout it will check if the cell in front and the cell to the left of the cell in front are empty to avoid collision. If the car in the roundabout is moving with low speed, 1, the car entering the system would enter the roundabout as long as other car is at least one cell away. If there is a car in the system that is two cells away and is speeding, 2, the car entering the system will wait for the speeding car to pass before entering the system.

3.7.4. Direction Ports

The direction ports are initialized at the start and never changed. They are used to help guide cars through the roundabout system.

3.7.5 isExit Ports

The isExitPorts are initialized at the start and never changed. They are used to indicate to cars if the adjacent cell is an exit cell. The car's exitNumberPort must match that of the cell's isExitPort for the car to leave the system.

3.7.6. Local neighbors

The local neighbor representation is shown below. This is different from our assignment two solution where positing tracking neighbours were used by the cells that are not near an exit so the correct rule will be executed and the car will moved to the correct position. This is not the case anymore.

```
type : cell
width : 8
height : 6
delay : transport
defaultDelayTime : 100
border : nonwrapped

neighbors : roundabout(-2,-2) roundabout(-2,-1) roundabout(-2,0)
roundabout(-2,1) roundabout(-2,2)

neighbors : roundabout(-1,-2) roundabout(-1,-1) roundabout(-1,0)
roundabout(-1,1) roundabout(-1,2)

neighbors : roundabout(0,-2)  roundabout(0,-1)  roundabout(0,0)
 roundabout(0,1)  roundabout(0,2)

neighbors : roundabout(1,-2)  roundabout(1,-1)  roundabout(1,0)
 roundabout(1,1)  roundabout(1,2)

neighbors : roundabout(2,-2)  roundabout(2,-1)  roundabout(2,0)
 roundabout(2,1)  roundabout(2,2)
```

3.7.6. Roundabout Ports

When a car leaves the mini roundabout to enter into another mini roundabout, information about the car is passed along. Information such as carExitNumber, numofhopsleft and speed are composed into 3 digit integer which is sent out of the respective port (out_west or out_east) to the next mini roundabout. The next roundabout will decompose this 3 digit integer and set the entrance cell's neighbor ports for carExitNumber, numofhopsleft, and speed.

3.7.7. Roundabout Rules

In order to replicate the behavior of the real HCR roundabout and car movements, different rules were developed. The rules are:

- Initial rules
- Movement rules (counter clockwise)
- Entering mini-roundabout exit rule
- Leave mini- roundabout rule
- Car entering cells
- Car exiting cells
- Yield rules
- Car exiting Hatton Round About system
- Car enter Hatton Round About system

The roundabout rule format is as follows:

```
<port_assignations>
[ <assignations> ]
 <delay>
 <condition>
```

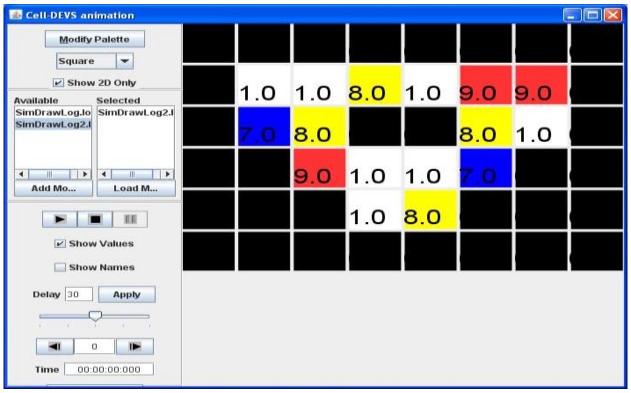
Example rule: The following rule is used for initializing the cells. This will be run only once by each cell at the beginning. It is used to transfer the information from the cell's state variables, that were previously determined in the roundabout.stvalues file, to the neighbor ports.

```
[roundabout-rule]
%////////////////////////////INITIALIZATION/////////////////////////
%initialization of cells (to be run only once by each cell)
rule : { ~directionport := $direction;
~isexitport := $isexit;
~exitnumberport := (0,0)~exitnumberport;
~numhopsleftport := 0;}
      { $initDone:=1;}
  100
  { $direction > 0 and
        $initDone =0 }
%//////////////////////END OF INITIALIZATION///////////////////////
```

4. SIMULATION RESULTS

The models and simulations were developed as discussed in the previous sections. Simulations were then run and the captured results were analyzed. In this section, the simulation and the results will be discussed.

Figure 5 below shows a test with an initial congested mini roundabout, which was captured in the CD++ builder. The cells that has blue (7), yellow (8) and red (9), has cars that are moving towards west, south and east exits respectively. The video that attached with the package shows the movements of these cars. It shows how cars yield the cars in the system before entering the roundabout. It also shows how the cars will exit when they arrive at their target exits.



Delay 30

0

Time 00:00:00:000

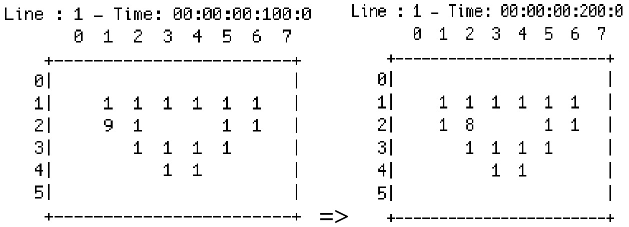
	1.0	1.0	8.0	1.0	9.0	9.0	
	7.0	8.0			8.0	1.0	
		9.0	1.0	1.0	7.0		
			1.0	8.0			

Figure 5: Initial congested mini-roundabout

As it was mentioned earlier, the east exit of one mini-round about is connected with west exit of its adjacent (right) roundabout. Same way, west exit of the same roundabout is connected with the east exit of its adjacent (left) roundabout. When the car leaves from one mini-roundabout to enter the next one, the car characteristics (exit number, number of hops left and the speed) will be sent to the next mini-roundabout through a port (out_west or out_east).

The following figures show different scenarios that were used for testing the mini-roundabout. It verified that model behavior meets the expected results.

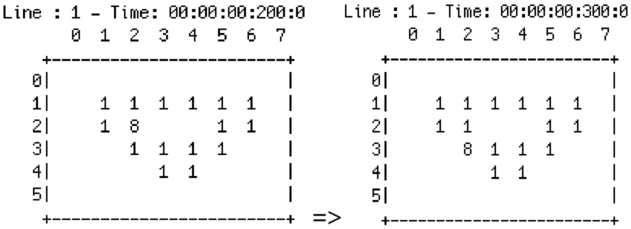
In first scenario it was tested that when the car enters the mini-roundabout from another mini-round and the 'numofhopsleftport' is at '0', the exit number gets updated to '8' indicating that in this mini roundabout the car will leave the system, at the south exit, and NOT entering into another mini roundabout. As you can see in the figure 6, the car with the exit number 9 enters the mini-roundabout from the left. According to the target HCR exit of the car, a new exit number of the current mini-roundabout will be assigned, which is 8 in this case.



```
Line : 1 - Time: 00:00:00:100:0  Line : 1 - Time: 00:00:00:200:0
     0  1  2  3  4  5  6  7           0  1  2  3  4  5  6  7
   +------------------------+       +------------------------+
  0|                        |      0|                        |
  1|    1  1  1  1  1  1    |      1|    1  1  1  1  1  1    |
  2|    9  1        1  1    |      2|    1  8        1  1    |
  3|       1  1  1  1       |      3|       1  1  1  1       |
  4|          1  1          |      4|          1  1          |
  5|                        |      5|                        |
   +------------------------+ =>    +------------------------+
```

Figure 6: Testing exit number update when car enters

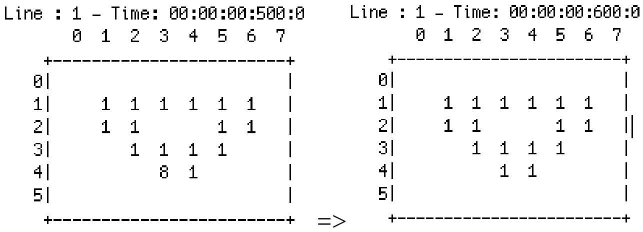
Car movement inside the mini-roundabout was tested as the next scenario. Figure 7 below illustrates that. The car with exist number 8 moves from the cell (2,2) to (2,3). The previous cell gets updated.



```
Line : 1 - Time: 00:00:00:200:0  Line : 1 - Time: 00:00:00:300:0
     0  1  2  3  4  5  6  7           0  1  2  3  4  5  6  7
   +------------------------+       +------------------------+
  0|                        |      0|                        |
  1|    1  1  1  1  1  1    |      1|    1  1  1  1  1  1    |
  2|    1  8        1  1    |      2|    1  1        1  1    |
  3|       1  1  1  1       |      3|       8  1  1  1       |
  4|          1  1          |      4|          1  1          |
  5|                        |      5|                        |
   +------------------------+ =>    +------------------------+
```

Figure 7: Testing Movement rules

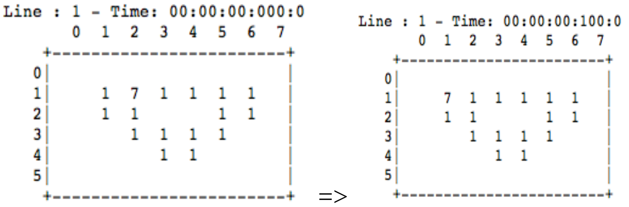
Car exiting the mini-roundabout was tested next. The figure 8 below shows how the car in the south exit leaves the roundabout that the cell is being replaced with 1.



```
Line : 1 - Time: 00:00:00:500:0  Line : 1 - Time: 00:00:00:600:0
     0  1  2  3  4  5  6  7           0  1  2  3  4  5  6  7
   +------------------------+       +------------------------+
  0|                        |      0|                        |
  1|    1  1  1  1  1  1    |      1|    1  1  1  1  1  1    |
  2|    1  1        1  1    |      2|    1  1        1  1    ||
  3|       1  1  1  1       |      3|       1  1  1  1       |
  4|          8  1          |      4|          1  1          |
  5|                        |      5|                        |
   +------------------------+ =>    +------------------------+
```

Figure 8: Testing exit rules (out of HCR)

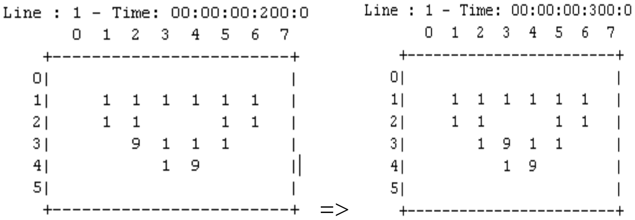
Figure 9 shows the scenario where a car approaches the mini-roundabout west exit and its ‘numberofhopsleft’ for the target exit attribute is decremented by one (not shown in diagram) when car leaves the exit. In this case, the information is sent to the ‘out_west’ port.



```
Line : 1 - Time: 00:00:00:000:0  Line : 1 - Time: 00:00:00:100:0
     0  1  2  3  4  5  6  7           0  1  2  3  4  5  6  7
   +------------------------+       +------------------------+
  0|                        |      0|                        |
  1|    1  7  1  1  1  1    |      1|    7  1  1  1  1  1    |
  2|    1  1        1  1    |      2|    1  1        1  1    |
  3|       1  1  1  1       |      3|       1  1  1  1       |
  4|          1  1          |      4|          1  1          |
  5|                        |      5|                        |
   +------------------------+ =>    +------------------------+
```

Figure 9: Testing exit into another mini-roundabout and number of hops left is decrement by one

The following test case is created to test whether the entering car factors in the speed of the car that is already in the system when it tries to enter the system. As you can see in the figure 10, in the first drawing, the car in the cell (4,4) waiting to enter the mini-round about. It checks the cell in front (4,3) and the cell on the left (3,3) to it to see if they are free. It senses that they are free. However it further checks that the car on the left in cell (2,3) with the carExitNumber 9 is moving with high speed(2). It decides not enter the exit to avoid collision as the other car will occupy the cell at the same time. It yields the car in the system and waits until the cell in front becomes free. This test was passed. If the speed of the car in the roundabout was slow (1), the car entering the roundabout would have entered.



```
Line : 1 - Time: 00:00:00:200:0  Line : 1 - Time: 00:00:00:300:0
     0  1  2  3  4  5  6  7           0  1  2  3  4  5  6  7
   +------------------------+       +------------------------+
  0|                        |      0|                        |
  1|    1  1  1  1  1  1    |      1|    1  1  1  1  1  1    |
  2|    1  1        1  1    |      2|    1  1        1  1    |
  3|       9  1  1  1       |      3|       1  9  1  1       |
  4|          1  9          ||     4|          1  9          |
  5|                        |      5|                        |
   +------------------------+ =>    +------------------------+
```

Figure 10: Testing entering car factors in the speed of the car in the system when deciding to enter the system

Example testing rule is shown below:

```
%%%%%TESTING-numhopsleftport%%%%%
%(2,1)
rule :    { ~exitnumberport := 9;
          ~numhopsleftport := 1;}
          {$testinitDone:=1; }
           100
          { (-1,0)~isexitport = 7 and
          $testinitDone =0}
```

Overall, extensive tests were done to ensure each components functions properly. Then they were integrated and integration tests were performed. Many different scenarios and different inputs were covered in the testing. The test results were satisfactory. We used RISE software with Lopez API for testing. RISE supports different CD++ versions of Cell-DEVS formalism. Lopez is an extended version of CD++ for allowing the cells to use multiple state variables and to use multiple ports for inter-cell communications. Since we used multiple state variables and multiple neighbor ports in our models, we used Lopez API.

Testing was initially performed on the workstation since the RISE server was down. Running these simulations on workstation consumes lot of time due to the fact of having issues trying to bring up the simulation server. Once the RISE web server was available, all of the testing was perform on the web server.

5. CONCLUSION

In this paper, CD++ Cell-DEVS and RISE software, lopez API, are used for simulation to reproduce the behavior of Hatton Cross Roundabout system and to further identify the factors that affect the throughput of the Hatton Cross Roundabout system which is a modern roundabout where large ring of road composed of six mini roundabouts. Cell-DEVS is an extension of DEVS that supports defining and executing cellular automata models.

The project was based on assignment two and expanded it to make the roundabout behavior more realist by considering more factors and scenarios. Issues from assignment 2 that were found were fixed in this project. Multiple states were introduced to factor speed, number of hops and etc.

For the testing, we considered many scenarios in multiple test cases and tested each component individually and extensively before combining them and testing to ensure each components functions properly. We used RISE software with Lopez API for testing. Testing and running the simulation of the RISE server was much faster than running on the workstation.

During development and testing phase, many challenges were encountered and much time was spent debugging the issues. One of the main challenges faced was the compiler doesn't give details about the syntax errors such as what the error was and where. Hours were spent in some cases in debugging the code before finding the actual syntax error. Another issue was that the RISE server was down for about a week therefore hindering testing and simulation of the code. Learning curve also played an important role in the slow start.

At the end, many issues were overcome and most of the goals that were set were achieved for this project. Due to time constrains, the last component of connecting mini-roundabout together was not achieved. Given more time, and more support with the tools, all of the project's goals would be fully achieved.

Overall, it was a very good learning experience of simulating a real system using DEVS, Cell_DEVS, RISE, and other tools.

6. REFERENCE

[1] G. Wainer; N. Giambiasi: "Application of the Cell- DEVS Paradigm for Cell Spaces Modeling and Simulation", *Simulation*, Vol. 71, No. 1, pp. 22-39, January 2001.

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

[3] http://www.airliners.net/aviation-forums/non_aviation/print.main?id=1075964
Accessed: December 15, 2012

[4] http://en.wikipedia.org/wiki/Roundabout
Accessed: December 15, 2012

[5] Gwizdalla, T.M.; Grzebielucha, S.; , "The traffic flow through different form of intersections," *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on* , vol., no., pp.299-304, 8-10 Oct. 2010

[6] J. Ameghino; G. Wainer: "Application of the Cell-Devs paradigm using N-CD++", *Summer computer simulation conference*, 2000

[7] Chaudhuri, P.P.; Chowdhury, D.R.; Paul, K.; Sikdar, B.; , "Theory and Applications of Cellular Automata for VLSI Design and Testing," *VLSI Design, 2000. Thirteenth International Conference on* , vol., no., pp.4, 2000

BIOGRAPHIES

PETER MIEBACH is a Masters of Engineering student at the Computer Systems Department at Carleton University, Ottawa, Canada. He completed his Bachelor's degree in Communications engineering from Carleton University in 2008. He has been working in private sector as software developer since then. His e- mail address is petermiebach@cmail.carleton.ca

PRADEEP GUNARATNAM has received Bachelor in Communications Engineering from the Carleton University, Ottawa, Canada in 2008. He has worked as software developer in private and public sectors. He is currently pursuing his Masters of Engineering in Electrical and Computer Engineering at Carleton University (expected to graduate in 2013). His e-mail address is pradeepkumargunaratn@cmail.carleton.ca