**CARLETON UNIVERSITY**

**SYSC-5104**

*ASSINGMENT-1*

**FOOD COURT SURVEY MODEL**

SHAGUN (101055437)

PRINCY (101055791)

# INDEX:

[A .Description of the Model](#_Toc57980161)

B. [Devs formalisation](#_Toc57980162)

C. [Cashier Model with Counter Lane Structure (coupled Model)](#_Toc57980163)

D. [Cashier Model with Counter Lane Formal Specification](#_Toc57980164)

E. [Atomic Models: Formal Specification,Real world Implementation & Unit Testing](#_Toc57980165)

F. [Ordering Station Model](#_Toc57980166)

G. [Cashier Model](#_Toc57980167)

H. [Counter Model](#_Toc57980168)

I. [Auditor Model](#_Toc57980169)

J. [Integration Testing Strategy](#_Toc57980170)

K. [Simulation Analysis & Examples](#_Toc57980171)

L. [COUNTER\_1](#_Toc57980176)

M. [COUNTER\_2](#_Toc57980173)

N. [COUNTER\_3](#_Toc57980174)

O. [FOODCOURT](#_Toc57980177)

P. [COUNTER\_COUPLE](#_Toc57980178)

Q. COUNTER1\_1…………………………………………………………………..

R. COUNTER\_0……………………………………………………………………

**PART 1 : DESCRIPTION**

In this model we are surveying a food court, the things which we are taking into consideration are:

a. Number of customers in the food court

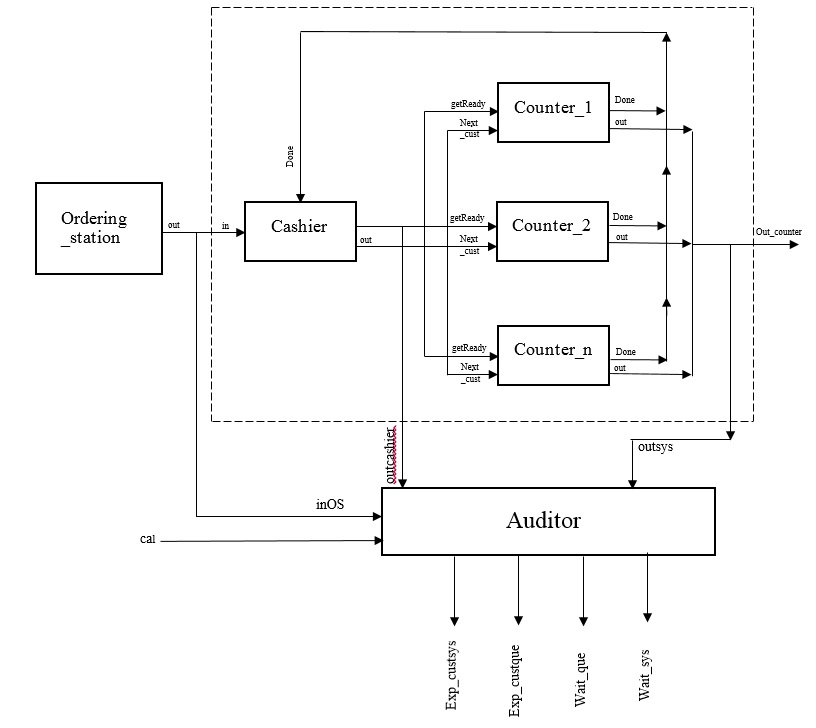
b. Waiting time for a customer in the queue before receiving the food order

c. Total customers standing in the queue waiting for their order.

d. Total time it took for the customer in the food court starting from the point where he ordered his food up until the point where he finally received his order.

We are building 7 atomic models and one couple model in the food court survey model.

**DIAGRAM:**



1. **ORDERING STATION MODEL**

In this model the time at which customer enters the food court is noted and calculation of total time taken by the food court to serve the customer is calculated. One thing to note here is that we are assuming that the customer is already there inside the ordering station hence we are not giving any input to the system and time gap between the arrivals of the next customer inside the ordering station is dependent on the configuration given by user.

1. **CASHIER MODEL**

Next atomic model is the cashier model, in this model the customers will be given the counter number where the food that they had ordered in the previous model will arrive. Here we are using the simple technique i.e. the counter number will be allocated first to the customer who arrives first to the cashier. In this model the customer will be directed to the counter in which his respective food will arrive.

1. **COUNTER MODEL**

Third atomic model is the counter model, in this model there are three counters that we have built, the decision in which counter the next customer will arrive to collect his food will be made by the cashier .This is basically the last process in the model and we will calculate total time spent by the customer in the food court from the point where he ordered his food this last step which is receiving food from the counter.

1. **AUDITOR MODEL**

Fourth atomic model is the most important one because in this model we are calculating everything, the final survey report of the food court is built through this model. Here the total time the customer waited in the queue before going to the counter to receive the food is calculated, this will include all the time taken by the cashier to make the decision before giving the signal to the customer to go to particular counter number to receive his order. Also here total number of customers in the food court is noted and total time taken by the food court to serve the customers.

a. exp\_custsys = Expected number of customers in food court.

b. exp\_custque= Expected number of customers in queue inside the cashier model

c.Wait\_que = Waiting time in food court.

d.Wait\_sys = Waiting time in queue.

Rest of all the models are the counters on which the customer’s food is arriving, the role of counter is just to serve the customer with the food, which customer ordered which food is taken care of by the cashier.

In case of coupled model we have coupled all the counter models together with the cashier model, in this there is one input from where all the customers will enter in order to go to respective counters to collect their food and there is one output from where all the customers will exit.

**PART 2 :**

# Formal Specification

## 

## Cashier Model with Counter lane Formal Specification (DEVS FORMALISATION)

#### **Specification:**

FSM = Food court survey model.

CounterX = all counters.

IC = Internal connections

EOC = External output connections.

EIC = External Input connection.

FSM-Foodcourtmodel

FSM = {X, Y, {order, cashier, {counter\_0, counter\_1, counter\_2, counter\_3}, auditor}, EIC, EOC, IC, Select}

X = {cal}.

Y = {out\_counter, wait\_que, wait\_sys,exp\_custque,exp\_custsys}.

IC = { (order.out, cashier.In),

(cashier.out, counter*X*.In),

(cashier.getReady, counter*X*.getReady),

(counterX.done, cashier.done),

(ordering\_station.out, auditor.inOS),

(cashier.out, auditor.outsys),

(counter*X*.out, auditor.outsys) }

EOC = {(counterX.out, FSM.out), (auditor.exp\_custsys, FSM.exp\_custsys),

(auditor.wait\_sys, FSM.wait\_sys), (auditor.exp\_custque, FSM.exp\_custque),

(auditor.wait\_que, FSM.wait\_que)}.

EIC = {(FSM.cal, auditor.cal)}

Select: (order, cashier, {counter\_0, counter\_1,counter\_2,counter\_3 }, auditor) = order.

Select: (cashier, {counter\_0, counter\_1,counter\_2,counter\_3}, auditor) = cashier.

Select: ({counter\_0, counter\_1,counter\_2,counter\_3}) = Either one.

Select: (auditor, any other block) = any other block.

## Atomic Models :

## 

## Formal Specification

### ORDERING STATION Model

#### **Specification :**

Order = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}

X = None.

Y = {out}.

S = {Active}.

EF = None.

IF = IF (Active) = Active.

OF (Active) = out.

TA (Active) = delay for distribution value.

#### **Implementation :**

Note that the distribution is configurable.

**InitFunction**:

{

Initialize all values.

Sigma = 0.

}

**InternalFunction**

{

If all customers were generated for previous time unit

Get the customers generated rate based on a distribution for this time unit.

If there is a customer needs to be generated in this time unit

Sigma = this customer interval time within this time unit.

Else

Sigma = time unit.

}

**OutputFunction**:

{

If there is a customer needs to be generated in this time unit

{

Increment the customer id by one.

Send the customer id on the “out” port.

Decrement the number of customers that need to be generated in this time unit.

}

}

### CASHIER Model :

#### **Specification :**

cashier = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}

X = {in, done}.

Y = {out, getReady}.

S = {(cashier.length >0 with done), (cashier.length >0 without done),

(cashier.length = 0 with done), (cashier.length = 0 without done}.

EF: EF ((cashier.length >0 with done), in) = (cashier.length >0 with done).

EF ((cashier.length >0 without done), in) = (cashier.length >0 without done).

EF ((cashier.length = 0 with done), in) = (cashier.length >0 with done).

EF ((cashier.length = 0 without done), in) = (cashier.length >0 without done).

EF ((cashier.length >0 with done), done) = (cashier.length >0 with done).

EF ((cashier.length >0 without done), done) = (cashier.length >0 with done).

EF ((cashier.length = 0 with done), done) = (cashier.length = 0 with done).

EF ((cashier.length = 0 without done), done) = (cashier.length =0 with done).

IF = IF (cashier.length >0 with done) =

(cashier.length = 0 with done). → If cashier.length = 1 and done > 1.

(cashier.length = 0 without done). → If cashier.length = 1 and last done .

(cashier.length > 0 with done). → If cashier.length >1 and done > 1.

(cashier.length > 0 without done). → If queue.length >1 and last done.

IF (any other case) = Passive.

OF (cashier.length >0 with done) = getReady and out.

TA (cashier.length >0 with done) = 0.

#### **Implementation :**

Note that the number of counters and counters Ids are configurable.

**InitFunction**:

{

Initialize requests list, request count and cashier elements list.

}

**ExternalFunction**:

{

If message arrived on the “in” port then

{

Push element to the back of the queue in cashier.

If this is the first element and there are requests then

Sigma = 0.

Else if there are no requests then

Passivate.

}

If message arrived on the “request” port and valid server Id then

{

Register server request.

Increment request count by one.

If this is the first request and there are elements in the queue then

Sigma = 0.

Else if there are no elements in the queue then

Passivate.

}

}

**InternalFunction**

{

Pop front element from the queue.

If there are more requests and elements in the queue then

Sigma = 0.

Else

Passivate.

}

**OutputFunction**:

{

Get counter Id.

Send the counter Id on the “getReady” port.

Send value on the “out” port.

}

### COUNTER Model :

#### **Specification :**

Server = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}

X = {next\_cust, getReady}.

Y = {out\_counter, request}.

S = {has\_done, processing\_service}.

EF: EF (has\_done, in) = processing\_service.

EF (has\_donet, getReady) = has\_done.

EF (processing\_service, in) = Not allowed.

EF (processing\_service, getReady) = Not allowed.

IF = IF (processing\_service) = Passive.

OF (has\_done) = out\_counter.

OF (processing\_service) = out\_counter, request

TA (processing\_service) = serving time based on allocation.

#### **Implementation :**

Counter Id and the distribution type are configured by the user.

**InitFunction**:

{

Force sending initial request to the queue in cashier model

}

**ExternalFunction**:

{

If passive and message is received on “getReady” port with my Id

{

Now I am “ready” for data.

}

Else if passive and message is received on “in” port and I am ready for data

{

Sigma = serving time based on a allocation.

}

}

**InternalFunction**

{

Now I am not “ready” for data.

Go to Passive state.

}

**OutputFunction**:

{

If was ready for data then

Output data on the “out” port.

Output my Id on the done port.

}

### AUDITOR Model :

#### **Specification :**

Performance = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}

X = {inOS, outcashier, outsys, cal}.

Y = {exp\_custsys, wait\_sys,exp\_custque , wait\_que}.

S = {Active, Passive}.

EF = EF (Active || Passive, cal) = Active.

EF (Active || Passive, inOS || outcashier || outsys) = Passive.

IF = IF (Active) = Passive.

OF (Active) = exp\_custsys, wait\_sys, exp\_custque,wait\_que.

TA (Active) = 0.

#### **Implementation :**

Please note that wait\_que and wait\_sys use “seconds” as the time unit.

**InitFunction**:

{

Initialize all data collection values and lists.

}

**ExternalFunction**:

{

If message is received from “inOS” port then

{

Count this customer.

Time stamp this customer arrival.

Calculate exp\_custsys and exp\_custque.

}

Else if message is received from “outcashier” port then

{

Count this customer.

Calculate exp\_custsys and wait\_sys.

}

Else if message is received from “outsys” port then

{

Count this customer.

Calculate exp\_custque and wait\_que.

}

Else if message is received from “cal” port then set Sigma to zero

}

**InternalFunction**

{

Passivate.

}

**OutputFunction**:

{

Output exp\_custsys, wait\_sys, exp\_custque and wait\_que.

}

#### **PART 3 :**

#### 

#### **UNIT TESTING :**

#### **Unit Testing for ORDERING STATION:**

1. We will run the simulation for a particular period of time.
2. We will generate the number based on each distribution:
   1. The customer id should represent the arrival rate of a distribution.
   2. The long-term mean should be closed to the provided distribution mean.
3. Generate test files: O\_S.bat, order\_station.ma and O\_S.out.

ORDERING STATION MODEL is working with the constant and poisson distribution but it will not work with the exponential distribution.

#### **Unit Testing for CASHIER MODEL:**

1. We have to test that the customer is being sent to the correct counter to take his food.
2. We will check the system for some incorrect value of the counter given to the customer.
3. Test the queue receiving a request from counter already placed a request.
4. Test the queue when it receives a request from other counter while servcing the another customer.
5. Test the queue in the cashier model when it receives a value while it is occupied.
6. We have to check for the order of the output.

Cashier testing related files: cashier.ma, cashier.bat,cashier.out and cashier.ev.

#### **Unit Testing for COUNTER MODEL** **(all counter models are coupled into one model):**

1. Test for the counter receiving protocol case: It should receive its first Id on the “getReady” port followed by a value on its “next\_cust” port.
2. Test if the counter receives other server Id on the “getReady” port followed by a value on the “next\_cust” port.
3. We have to check if the counter receives a value on the “next\_cust” port without receiving anything on the “getReady” port first.
4. Test the service time if it follows the provided distribution.

Counter testing files:counter.ma, counter.bat, counter.out and counter.ev

#### **Unit Testing for the AUDITOR MODEL:**

We will feed the model with a number of messages in its ports: inOS, outcashier and outsys, and then verify the output values of wait\_que, exp\_custque, exp\_custsys and wait\_sys against the calculated values by hand .The above values will be sent when the model will receive the value in its cal port. We have used ‘second’ as the time unit for wait\_que and wait\_sys

Performance test files: auditor.bat, auditor.ma, auditor.ev and auditor.out.

# Testing Strategy

A testing was performed for each model as explained in the above Unit Testing sections in the following sequence:

1. Ordering station
2. Cashier
3. Counter
4. Auditor.
5. Food court survey model

1. Each model of the above has its test cases in “modelName.ev” file.

2. Its structure in “modelName.ma” file.

# Simulation Analysis & Examples

Let:

Arrival rate = λ, Processing rate = µ, Probability =P

## Counter1

**Counter1** with the following parameters:

* Single server/Single queue
* λ = 3 customers/per time unit
* µ = 8 customers/per time unit

**Analytic method**:

exp\_custsys = λ 2 / µ(µ- λ).

Wait\_sys = λ / µ(µ- λ).

Exp\_custque = λ / (µ- λ).

Wait\_que = wait\_sys/ λ.

Exp\_custsys = 0.2. wait\_sys = 0.6. exp\_custque = 0.225. Wait\_que = 0.075.

**Simulation Method:**

This model exists in (counter1.ma) file.

The generator is configured with Poisson distribution with mean of 3 and the counter is configured with Poisson distribution with mean 8.

After running the simulation (counter0.bat) for 5 minutes, the following results were obtained (foodcourt.out):

00:05:00:000 exp\_custsys 0.557951

00:05:00:000 exp\_custque 0.0216764

00:05:00:000 wait\_que 0.763215

00:05:00:000 wait\_sys 0.168861

## Counter2

Counter**2** with the following parameters:

* Two servers/Single queue
* λ = 35 customers/per time unit
* µ = 25 customers/per time unit

**Analytic method**:

Exp\_custsys = ((λ/µ) P (n >= s)) / (1-(λ/µ)). (c = # of counters)

Exp\_custque = exp\_custsys / λ.

Wait\_que = exp\_custsys + λ/µ.

wait\_sys = Wait\_que / λ.

wait\_sys = 0.064. Ls = 2.263. exp\_custsys = 0.863. Exp\_custque = 0.0248.

**Simulation Method:**

This model exists in (counter2.ma) file.

The generator is configured with Poisson distribution with mean of 35 and the two servers are configured with Poisson distribution with mean 25.

After running the simulation for 5 minutes, the following results were obtained (foodcourt.out):

00:05:00:000 exp\_custsys 0.567269

00:05:00:000 Exp\_custque 0.0288511

00:05:00:000 Wait\_que 1.69845

00:05:00:000 wait\_sys 0.0442661

## Counter3

**Counter3** with the following parameters:

* Three servers/Single queue
* λ = 54 customers/per time unit
* µ = 20 customers/per time unit

**Analytic method**:

exp\_custsys = ((λ/µ) P (n >= s)) / (1-(λ/µ)). (c = # of counters)

Exp\_custque = exp\_custsys / λ.

Wait\_que = exp\_custsys + λ/µ.

wait\_sys = Wait\_que / λ.

wait\_sys = 0.1528. Wait\_que = 8.254. exp\_custsys = 7.354. Exp\_custque = 0.1362.

**Simulation Method:**

This model exists in (Counter3.ma) file.

The generator is configured with Poisson distribution with mean of 54 and the three servers are configured with Poisson distribution with mean 20.

After running the simulation for 5 minutes, the following results were obtained (foodcourt.out):

00:05:00:000 exp\_custsys 13.4632

00:05:00:000 Exp\_custque 0.233314

00:05:00:000 Wait\_que 16.322

00:05:00:000 wait\_sys 0.285632

## foodcourt

**foodcourt** with the following parameters:

* Four servers/Single queue
* λ = 100 customers/per time unit
* µ = 30 customers/per time unit

**Analytic method**:

exp\_custsys = ((λ/µ) P (n >= s)) / (1-(λ/µ)). (S = # of servers)

Exp\_custque = exp\_custsys / λ.

Wait\_que = exp\_custsys + λ/µ.

wait\_sys = Wait\_que / λ.

Wait\_sys = 0.0662. wait\_que = 6.62. exp\_custsys = 3.29. Exp\_custque = 0.0329.

**Simulation Method:**

This model exists in (foodcourt.ma) file.

The generator is configured with Poisson distribution with mean of 100 and the four counters are configured with Poisson distribution with mean 30.

After running the simulation for 5 minutes, the following results were obtained :

00:05:00:000 exp\_custsys 2.11242

00:05:00:000 Exp\_custque 0.0154731

00:05:00:000 Wait\_que 5.4825

00:05:00:000 wait\_sys 0.0496017

## Counter1\_1

This example is about the deterministic

**Counter1\_1** with the following parameters:

* Single server/Single queue
* λ = 5 customers/per time unit
* µ = 8 customers/per time unit

**Analytic method**:

exp\_custsys = λ 2/2µ(µ- λ).

Exp\_custque = exp\_custsys /λ.

exp\_custsys = 0.5208. Exp\_custque = 0.104.

**Simulation Method:**

This model exists in (Counter1\_1.ma) file.

The generator is configured with Poisson distribution with mean of 5 and the server is configured with Constant distribution with value 8.

After running the simulation (Counter1\_1.bat) for 5 minutes, the following results were obtained:

00:05:00:000 exp\_custsys 0.649786

00:05:00:000 Exp\_custque 0.0283375

00:05:00:000 Wait\_que 0.953947

00:05:00:000 wait\_sys 0.153356

## counter0

**counter0** with the following parameters:

* Single server/Single queue
* λ = 5 customers/per time unit
* µ = 8 customers/per time unit

**Simulation Method:**

This model exists in (counter0.ma) file.

The generator is configured with Constant distribution with mean of 5 and the counter is configured with Constant distribution with value 8.

After running the simulation 5 minutes, the following results were obtained :

00:05:00:000 exp\_custsys 0.499833

00:05:00:000 Exp\_custque 0

00:05:00:000 Wait\_que 0.5

00:05:00:000 wait\_sys 0.125

## Counter\_couple

**Counter\_couple** with the following parameters:

* Four server/Single queue
* λ = 30 customers/per time unit
* µ = 8 customers/per time unit

.

**Simulation Method:**

This model exists in (Counter\_couple.ma) file.

The generator is configured with Constant distribution with mean of 30 and the four counters are configured with Constant distribution with value 8.

After running the simulation for 5 minutes, the following results were obtained (foodcourt.out):

00:05:00:000 exp\_custsys 6.82222

00:05:00:000 Exp\_custque 0.207972

00:05:00:000 Wait\_que 10.5579

00:05:00:000 wait\_sys 0.332763