Department of Systems and Computer Engineering

SYSC 5104 - Methodologies for Discrete Event Modelling and Simulation

Modeling a Food Bank using DEVS in CD++

Assignment 1

Part 1

Name: Emad Khan

Student ID: 101006749

Email: emad.khan@cmail.carleton.ca

Table of Contents

PART I

Food Bank Conceptual Model

A food bank is a non-profit organization that collects and distributes food to hunger relief charities. Food usually comes from various sources in the food industry, like grocery stores and wholesalers that have thousands of pounds of food to give away - food that could otherwise be thrown away. Food banks are found in most larger communities and rely on donors and volunteers to carry out day-to-day operations. Food banks in the U.S. are very diverse -- from small operations serving people spread out across large rural areas to very large facilities that store and distribute many millions of pounds of food each year, and everything in between. A variety of factors impact how food banks work, from the size of the facility to the number of active volunteers and donors.

The following is a simplified model to simulate the operation of a food bank, a place where stocks of food, typically basic provisions and non-perishable items, are supplied free of charge to people in need. Discrete event simulation allows us to evaluate the operating performance before the implementation of a system. In order to simulate the integration and interoperation between clients and the food bank, an operation management model is built using the DEVS formalism.

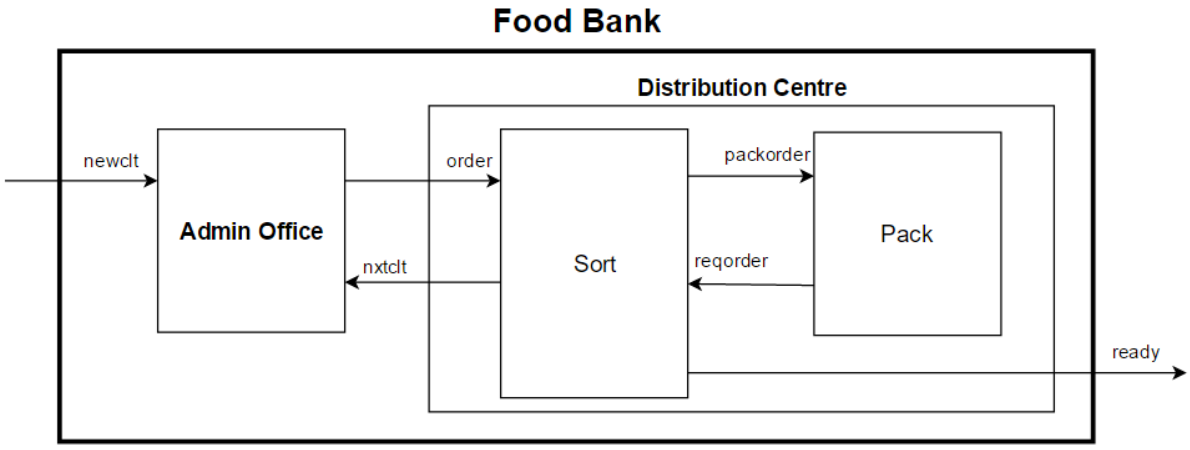


Figure 1: Food Bank Model

Model Description

The goal of the model will be to simulate the smooth operation of the food bank so that the clients receive their food. The food bank model is composed of two components: an atomic model called the ***Administration Office*** the simulates an office receiving clients and donations and performing administrative tasks, another atomic model called the ***Distribution Centre*** that simulates the process of preparing the food. The Distribution Centre component includes two atomic models, i.e. ***Sorting*** model represents sorting of the food and a ***Packaging*** model represents the packaging of the sorted food items. This conceptual model is illustrated in Fig 1.

Atomic Model: Admin Office

The atomic model Admin Office has two inputs;

- *newclt* - New clients visiting the food bank to take food
- *nextclt* - Input from the Sorting model requesting for the next client

It also has an output:

1. *order* – the output from the *Admin Office* creating an order for food

The atomic model *Admin Office* represents a queue model where clients arrive and arranged in a queue based on FIFO. The clients wait for their turn to be processed for food packages. All clients that arrive inform the admin office the size of their household. An order is then created and sent to the *Distribution Centre* after which the *Admin Office* waits for the input *nexclt* to create a new order. The operation hours of the food bank are from 9:00AM to 13:00PM.

Atomic Model: Distribution Centre

This atomic model consists of two subatomic models; *Sorting* and *Packaging*.

The subatomic model represents sorting out the food items for the client. *Sorting* has two inputs: *order* (from *Admin Office*) and *reqorder* (*from Packaging*). The input *reqorder* is received to send another order for packaging. There are also two outputs, one for *Packaging* and the other is the final output of the food bank. The output *packorder* is a request for *Packaging* subatomic model to start packing the food items. The output *ready* is also the output of the *Distribution Centre* and the Food Bank. This is output when the food is ready to be picked up by the clients.

The subatomic model Packaging represents packaging of food item. If the clients inform the admin office that they want food for a big household, then the sorted food items are sent in for packaging so that it's easier for the client to carry them. Packaging is costly and is therefore not done for all clients.

Part II

Model Specification & Testing Strategies

The food bank is modeled as a two-level DEVS model with one components that is described as coupled model and one components that is described as an atomic model. The DEVS formal specifications for each model is outlined below starting from atomic models and concluding with the food bank itself.

The discrete event system specification (DEVS) for the atomic models are defined as follows

Atomic Model: Admin

$$\text{Atomic Admin} = < X, Y, S, \delta ext, \delta int, \lambda, ta >$$

InPorts = {"newclt", "nxtclt@sort"}
X = {(p,v)|p∈InPorts, v∈Xp} is the set of input ports and values
S = {client seating, passive}
OutPorts = {order@sort}
Y = {(p,v)|p∈OutPorts, v∈Yp } is the set of Output ports and values
Internal Transition Function
δint (e) *{*

sigma = oo;
phase = passive ;

}

External Transition Function
δext (Client, Pack. reqorder){

case port
newclt {
sigma = servingtime

phase = servingClient
λ = internal queue, order@sort
}
sort.ready {
sigma = servingtime;
phase = servingClient;
λ = order@sort
}
end-case

ta = time advance

Output Function

λ = (if newcust {
if only order {order@sort}
else{queue}
if ready@sort {
if queue is not empty{order@sort;}
)

Atomic Model: Sort

Atomic Sort = $< X, Y, S, \delta ext, \delta int, \lambda, ta >$

InPorts = {Pack. reqorder, order}

X = {(p,v)|p∈InPorts, v∈Xp} is the set of input ports and values

S = {checking, passive}

OutPorts = {packorder, ready}

Y = {(p,v)|p∈OutPorts, v∈Yp } is the set of Output ports and values

External Transition Function

δext (client, Pack. reqorder) {
case port
admin.order {
sigma = checktime
cltcount = order.msg.value()
phase = checking
λ =packorder@pack
}
pack.reqorder{
sigma = Checktime;
phase = Checking;
λ = packorder@pack,next@admin

end-case

}

Internal Transition Function

δint (e) {

sigma = 0;

phase = passive;

}

Output Function

λ *= (if port.reqorder@pack {*

if cltcount > desired household{next@admin;}

else

if port.admin@order{packorder@pack;}

)

Atomic Model: Packaging

Atomic Pack = $< X, Y, S, \delta ext, \delta int, \lambda, ta >$

InPorts = {packorder}

X = {(p,v)|p∈InPorts, v∈Xp} is the set of input ports and values

S = {packing, passive}

OutPorts = {reqorder}

Y = {(p,v)|p∈OutPorts, v∈Yp } is the set of Output ports and values

External Transition Function

δext (e) *{*

assert port = packorder

sigma = packing time;

phase = packing;

lamda = reqorder@sort

}

Internal Transition Function

δint *(e, sigma) {*

phase = passive;

sigma = 0;

}

Coupled Model: Distribution Centre

$$\text{Coupled DistributionCentre} = < X, Y, D, \{Md \mid d \in D\}, EIC, EOC, IC, Select >$$

Input Set
X = {in}
Output Set
Y = {out}
Set of Component Names
D = {Sort, Pack}
DEVS Models
M = {M_{Sort}, M_{Pack}}
EIC = {(Sort, "order"), (Self, "in")}
External Output Couplings
EOC = {(Sort, "ready"), (Sort, "nxtclt"), (Self, "out")}
Internal Couplings
IC = {((Sort, "packorder"), (Sort, "reqorder")), ((Pack, "packorder"), (Pack, "reqorder"))}
Tiebreaker Function
Select: ({Pack, Sort})

Coupled Model: Food Bank

$$\text{Coupled FoodBank} = < X, Y, D, \{Md \mid d \in D\}, EIC, EOC, IC, Select >$$

Input Set
X = {in}
Output Set
Y = {out}
Set of Component Names
D = {Admin, Sort, Pack}
DEVS Models
M = { M_{Admin}, M_{Sort}, M_{Pack}}
External Input Couplings
EIC = {(Admin, "newclt"), (Self, "in")}
External Output Couplings
EOC = {(Sort, "ready"), (Self, "out")}
Internal Couplings
IC = {((Admin, "order"), (Admin, nxtclt)),((Sort, "order"), (Sort, "nextclt"), (Sort, "packorder"), (Sort, reqorder)), ((Pack, "packorder"), (Pack, "reqorder"))}
Tiebreaker Function
Select: ({Admin, Pack, Sort})

Part III

Testing Strategy

Atomic Model: Admin

Admin represents a queue for clients coming in on FIFO basis. The queue size is 10, so a memory is kept of all the clients walking into the food bank. Admin takes two inputs so it takes care of the input from the Distribution Centre to place a new order, and also ‘ready’ messages from Sort. The atomic model simultaneously keeps check of both the inputs.

Atomic Model: Sort

The atomic model sort represents sorting of food items for the client. It simultaneously receives both orders from the *admin*, and *reqorder* requests from Pack. On *reqorder* inputs from Pack, Sort will be tested that it correctly counts the order request from input *reqorder*. Inputs from *admin*, the atomic model *Sort* will be tested that it correctly starts sorting food once the order is received.

Atomic Model: Packing

Pack is similar to sort and it is also timed. It receives *packorder* inputs, increasing the total item to process an order since this is an additional task. When the packing is completed, it outputs reqorder and waits for the next one.

Coupled Model: Distribution Center

Distribution Centre is the coupled model consisting of Sort and Pack components. It receives orders, and informs when processing is completed. DS starts its process when it receives a orders, completes the order and outputs.

Coupled Model: Food Bank

Food Bank is the complete hierarchically coupled system, including Admin, Sort and Pack components. It receives clients and queues them. If the Distribution Centre is not busy, it passes queued clients to the Distribution Centre when he outputs *ready*. It must be able to start when it receives a client, send a 'first client in queue' to the Distribution Centre, and queuing others that arrive, and manage open close times for the shop.

Testing Set

1. Operation of the FoodBank model with inputs that represent a realistic simulation
2. Operation of the FoodBank model with extreme situations where inputs x - preset 0 transition times
3. Operation of the FoodBank model with extreme situations where inputs x - preset long transition times

Testing & Simulations of Models

Atomic Model: Admin

The model was tested by giving inputs of clients and then looking at the output to see if the clients were served on time. The admin model having two inputs; 'newcust' representing a new client coming and 'nxtclt' donating that another client is requested. To input 'nxtclt' is generated by the model Sort. Since this is independent testing of the admin, we assume that an input of `1` is being received after every 15mins. Please note, this is just being done to test this model and the original operation of the complete doesn't assume this.

The operational timing and preparation timing of the Admin Office are defined in '*admin.ma*'. For this test, we assume that the Admin Office requires 10mins to check identification documents and ask for household size.

Specifications for Model Admin:

```
[admin]
numberofSeats : 10
prepTime : 00:10:00:000
openHrs : 09:00:00:000
closeHrs : 13:00:00:000
```

Events for Model Admin:

The inputs are given in the file 'admin.ev'. An example with results is given below:

```
08:45:00:00 newclt 40   <-------- A
09:00:00:00 newclt 20   <-------- B
09:15:00:00 nxtclt 1    <-------- C
09:20:00:00 newclt 50
09:30:00:00 nxtclt 1
09:35:00:00 newclt 40
09:40:00:00 newclt 25
09:45:00:00 nxtclt 1
09:55:00:00 newclt 70
10:00:00:00 nxtclt 1
10:00:00:00 newclt 10
10:05:00:00 newclt 30
10:15:00:00 nxtclt 1
10:30:00:00 nxtclt 1
10:45:00:00 nxtclt 1
```

```
10:50:00:00 newclt 45
10:55:00:00 newclt 20
```

A. At 8:45 AM a client arrives at the Admin Office outside the operational hours.
B. The first client during the operational hours.
C. Input 'nxtclt' from Sort generated after every 15 mins.

Output for Model Admin:

```
09:05:00:000 order 20 <-------- B
09:25:00:000 order 50
09:40:00:000 order 40
09:50:00:000 order 25
10:05:00:000 order 70
10:20:00:000 order 10
10:35:00:000 order 30
10:55:00:000 order 45
11:05:00:000 order 20
```

A. Client 40 is not served since the Food Bank is closed and not operational.
B. We also see that Client 20 that arrived at 9:00AM is served after 5mins at 9:05AM
C. Since the next model is ready it keeps on asking for clients and all clients are served on time.

Atomic Model: Sort

The testing of atomic model Sort is similar to that of model admin. The model Sort takes output 'order' from Admin as input. To test this model, I have created some test inputs in the file 'sort.ev' which will have inputs at different times. The preparation time required to do the food sorting has been set at 1min and 50 seconds.

Specifications for Model Sort:

```
[sort]
prepTime : 0:01:50:000
```

Events for Model Sort:

```
00:00:00:00 order 3      <-------- A
00:02:00:00 reqorder 0   <-------- B
00:04:00:00 reqorder 0
00:06:00:00 reqorder 0
00:08:00:00 reqorder 0
00:08:05:00 order 5      <-------- C
00:08:00:00 reqorder 0
00:10:00:00 reqorder 0
00:12:00:00 reqorder 0
00:14:00:00 reqorder 0
```

```
00:16:00:00 reqorder 0
00:18:00:00 reqorder 0
00:20:05:00 order 2
00:22:00:00 reqorder 0
00:24:00:00 reqorder 0
00:25:00:00 order 3    <-------- D
00:26:00:00 reqorder 0
00:27:00:00 reqorder 0
```

A. First input order has the value 3, which means it has 3 members in its household and requires food packaging for all three members.
B. Input reqorder from Pack means that the packing has been completed.
C. Second Order is sent as Order 5
D. Order 3 is sent before Order 2 is completed to see how the model reacts.

Outputs for Model Sort:

```
00:01:50:000 packorder 1 <-------- A
00:03:50:000 packorder 2
00:05:50:000 packorder 3
00:07:50:000 ready 3     <-------- B
00:09:55:000 packorder 1
00:11:50:000 packorder 2
00:13:50:000 packorder 3
00:15:50:000 packorder 4
00:17:50:000 packorder 5
00:19:50:000 ready 5     <-------- C
00:21:55:000 packorder 1
00:23:50:000 packorder 2
00:25:50:000 ready 2
                         <-------- D
```

A. First Packing Order was created after 1min and 50 seconds of ordering as expected.
B. The whole order for the client was completed in 7 mins and 50 seconds.
C. Second Packing Order was also completed at the right time.
D. Order 3 which was sent in between the previous was not completed since the atomic model Pack can only handle one order at a time. Even though a request for the order was created but the since the previous order was not completed, it cannot take on a new order.

The model Sort functioned as expected. It was seen that it would not take on double orders and would also first give priority to previous orders. It serves only one order at a time.

Atomic Model: Pack

This is a simple model that must create an output after a certain amount of time once it receives the input. For our testing purposes, I have set the packing time to 2mins and 40 seconds. The input comes from the atomic model Sort and represents a packing order. This food items after sorting are packed and output as reqorder. The input values are saved in the file 'pack.ev'

Specifications for Model Pack:

```
[sort]
prepTime : 0:02:40:000
```

Events for Model Sort:

```
00:00:00:000 packorder 5 <---- A
00:03:00:000 packorder 2 <---- B
00:06:00:000 packorder 4
00:09:00:000 packorder 7
00:12:00:000 packorder 9
00:15:00:000 packorder 1
00:17:00:000 packorder 3 <---- C
00:18:00:000 packorder 1
00:20:00:000 packorder 3
00:23:00:000 packorder 4
00:25:00:000 packorder 3
00:26:00:000 packorder 5
```

A. First packing order sent at time zero.
B. Second packing order send after 3 minutes
C. A packing order sent before the previous order is completed, i.e, before the completion of the preparation time of 2minutes and 40 seconds.

Outputs for Model Sort:

```
00:02:40:000 reqorder 1 <---- A
00:05:40:000 reqorder 1 <---- B
00:08:40:000 reqorder 1
00:11:40:000 reqorder 1
00:14:40:000 reqorder 1
00:17:40:000 reqorder 1 <---- C
00:20:40:000 reqorder 1
00:25:40:000 reqorder 1
00:28:40:000 reqorder 1
```

A. First packing order received after 2mins and 40 seconds at the expected time.
B. Second packing order was received at the right time since it was sent after the previous one was completed.
C. Order here was rejected and not queued since it was when the atomic model Pack was active.

Coupled Model: Distribution Centre

The coupled model Distribution Centre consists of subatomic models 'Sort' and 'Pack' which I have tested independently. Coupling them together they must work similarly as they worked independently. The distribution centre again represents the sorting and packaging of the food based on the client's household members. A client with a big family size will eventually require more time since each individual in his family will need food packaged and ready.

To test this model, I have put in some sample inputs to test its behavior when both models are coupled together. The inputs are provided in the file 'DistributionCentre.ev' and it has input values representing the members in the household.

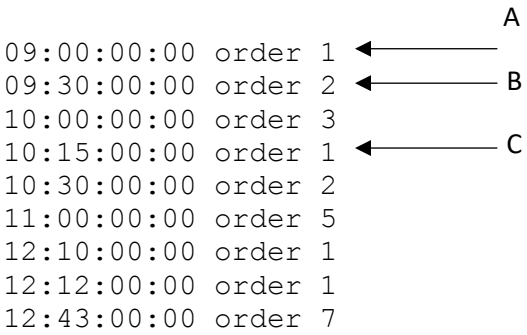
Specifications for Model DistributionCentre:

Setting the time for subatomic model Sort = 2 minutes to sort the food items.
Setting the time for subatomic model Pack = 6 minutes to pack the sorted food items.
This can be found in the file 'DistributionCentre.ma'

```
 [sort]
prepTime : 00:02:00:000

[pack]
prepTime : 00:06:00:000
```

Events for Model DistributionCentre:



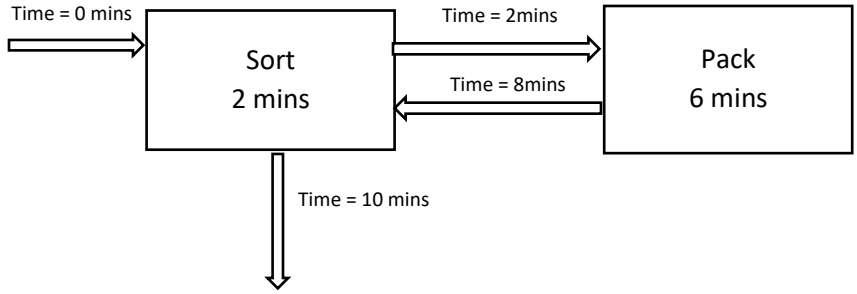
A. The first input represents a client order at 9:00AM and has only a single member in his family
B. The second input represents a client order having a two members in his family.
C. This client order arrives at a time when the previous order has not been completed

Outputs for Model DistributionCentre:

The outputs are in the file 'DistributionCentre.out'

```
09:10:00:000 ready 1 <---- A
09:48:00:000 ready 2 <---- B
10:26:00:000 ready 3
10:48:00:000 ready 2 <---- C
11:42:00:000 ready 5
12:20:00:000 ready 1
13:41:00:000 ready 7
```

A. The output is received after a total of 10mins



B. The output is received after 2+6+2+6+2 = 18 mins.
C. This pack order arrived when the DistributionCentre was active and busy with the previous order, hence it was rejected and cannot be seen in the output. The next order arrives when the model is passive is therefore processed normally.

Coupled Model: Food Bank

This coupled model resembles the final working model of a food bank. It consists of the atomic model Admin and the coupled model DistributionCentre. Atomic model admin functions as a queue model for clients. So if two clients arrive at the same time, it must queue on FIFO basis.

Specifications for Model DistributionCentre:

```
[admin]
numberofSeats : 10
prepTime : 00:05:00:000
openHrs : 09:00:00:000
closeHrs : 13:00:00:000

[sort]
prepTime : 00:02:00:000

[pack]
prepTime : 00:01:00:000
```

To test the final Food Bank model, I have set its operational timings from 9:00AM to 1:00PM. I have also set the number of seats that the queue can hold to 10. It takes 5 minutes for the Administration Office to complete its tasks. An order is then forwarded to the Distribution Centre which is a coupled model consisting of subatomic model Sort, that requires 2 minutes to sort the food items, and subatomic model Pack which needs 1 minute to pack the food items. The outputs are in the file 'FoodBank.ev'

Events for Model FoodBank:

```
08:50:00:00 newclt 4   <-- A
09:00:00:00 newclt 1   <-- B
09:15:00:00 newclt 2
09:40:00:00 newclt 3
10:00:00:00 newclt 7
10:33:00:00 newclt 4
10:51:00:00 newclt 9
11:23:00:00 newclt 5   <-- C
11:30:00:00 newclt 4   <-- D
11:32:10:00 newclt 8   <-- D
12:45:00:00 newclt 6
13:12:00:00 newclt 2   <-- E
```

A. A client arrives before the Opening Time of the Food Bank
B. The first client arrives having a household size of one member only
C. A client arrives before the previous client has been served
D. Multiple clients arrive before the previous client has been served
E. Client arrives after the Closing Time of the Food Bank

Events for Model FoodBank:

```
                          <-- A
09:10:00:000 ready 1      <-- B
09:28:00:000 ready 2
09:56:00:000 ready 3
10:28:00:000 ready 7
10:52:00:000 ready 4
11:26:00:000 ready 9
11:48:00:000 ready 5      <-- C
12:07:00:000 ready 4      <-- D
12:38:00:000 ready 8      <-- D
13:10:00:000 ready 6
                          <-- E
```

A. Client wasn't served since the Food Bank didn't open at the time.
B. First client was served in 10 minutes (Admin -> Sort -> Pack -> Sort -> Ready)
C. Client was queued even though the model was active and busy
D. Multiple clients all were queued and served once the previous was completed
E. Client wasn't served since the Food Bank is now closed.