**SYSC 5104 METHODOLOGIES FOR DISCRETE EVENT MODELLING AND SIMULATION**

**Assignment 1 – Part I - Conceptual Model – Block Diagram**

Yasser Jafer

Ottawa University – Student Number: 2554284

[yjafe089@uottawa.ca](mailto:yjafe089@uottawa.ca)

Technical Support Financial Dept

Validation

Coverage Plan Manager

Financial Queue

Financial Manager

Financial Q Control

Release Next

Request

Done

Plan OK

Result

Problem

In

Call in

Call in

Problem

**Part I – Conceptual Model (Revised)**

This Model simulates a model of simple financial department of a technical support call center. The model was revised based on a feedback from the instructor and the name of the model was changed to focus merely on handling financial transactions of a technical support call center.

This is a two level model which includes:

1. One coupled model (i.e. Financial Manager) and its 2 atomic sub models (namely Coverage Plan Manager and Validation).
2. Two atomic models (namely Financial Q and Financial Q Control).

The top model (Technical Support Financial Dept) has one input which are the calls made by customer who have technical problems with their systems. There are two final results associated with this model: Either the financial transaction was done successfully which means the caller has a valid coverage plan and his/her call is ready to be transferred to the technical support queue or the validation failed and the customer needs to disconnect and call later. So the Technical Support Q and Help Desk agent which were simply model of delay queue and were very similar to the Financial Q in their functionality were removed and new Atomic model i.e. Financial Q Control was added instead to control the Financial Q in a more appropriate way.

Description of each of the Coupled and Atomic models is explained below:

**Part II**

**Coupled Models:**

**Top Model:**

**Technical Support Financial Dept:** The top model representing the operational model for a simple Technical Support Financial Dept of a Technical Support Call Center

Input Ports: CALL\_IN

Output Ports: PROBLEM, DONE

Problem

Technical Support Financial Dept

Done

Call In

TSFD Formal Specifications:

TSFD = <X, Y, D, EIC, EOC, IC, SELECT>

X = <Call\_in>

Y = <Done, Problem>

D = <Financial\_Queue, Financial\_Manager, Financial\_Q\_Control>

EIC = {TSFD.Call\_in, Finanical\_Q.Call\_in}

EOC = {(Financial\_Manager.Done, TSFD.Done),

(Financial\_Manager.Problem.TSFD.Problem)}

IC = { (Financial\_Q.Control\_Out, Financial\_Q.Release\_Next),

(Financial\_Q.Call\_out, Finanical\_Manager.Call\_In),

(Financial\_Manager.out\_Plan\_Ok, Financial\_Control\_Q.in),

(Financial\_Manager.out\_Problem, Financial\_Control\_Q.in)

}

SELECT =Descending: {Financial\_Q\_Control, Financial\_Q, Financial\_Manager}

**Atomic Models:**

1. **Financial Q**: This is a model of queue with waiting calls, it takes inputs as calls and passes those calls after receiving control input through Release Next port which releases next waiting call in the Financial Q. The call will gets sent to the next model which is Financial Manager to process the financial transaction and required arrangements previous call is done. If new call gets added to the queue it will wait until the previous call has left the queue in FIFO characteristics.

Call In

Call Out

Financial Queue

Release Next

Input Ports: CALL\_IN

Output Ports: CALL\_OUT

Formal Specifications for Financial Queue = <X, Y, S, δint, δext, λ, ta>

X = {(“Call\_In”, integer), (“Release\_Next”, {1})}

Y = {“Call out”, integer}

S = {Phase, sigma, num\_waiting\_calls}

δext (Phase, sigma, num\_waiting\_calls, e, x){

Case Phase {

Passive:

Case Port {

Call\_In:

num\_waiting\_calls = num\_waiting\_calls + 1;

Release\_Next:

Phase = Active;

If num\_waiting\_calls = 0;

sigma = infinity;

else

sigma = Processing\_Time;

}

Active:

Case Port {

Call\_In:

If num\_waiting\_calls = 0

sigma = Prcossesing\_Time;

num\_waiting\_calls = num\_waiting\_calls + 1;

Release\_Next:

Ignore; Next call cannot be released unless the previous call has already left the financial agent phase.

}

}

δint (Phase, sigma, num\_waiting\_calls)

{

Case Phase {

Passive:

Ignore, Not Applicable;

Active:

Phase = passive;

num\_waiting\_calls = num\_waiting\_calls – 1;

sigma = infinity;

}

}

λ(Phase, sigma, num\_waiting\_calls)

{

Case Phase {

Passive:

Ignore, Not Applicable;

Active:

If (num\_waiting\_calls > 0)

Out = first waiting call in the queue

}

}

1. **Financial Q Control:** This model is responsible for controlling the Financial Q model. The output of Financial Manager coupled model will be sent to Financial Q Control via Control\_In port and Control\_Out message gets sent out to the Financial Q which will on its behalf release next waiting call in the queue.

Financial Q Control

Control Out

Control In

Input Port: Control\_In

Output Port: Control\_Out

Formal Specification for Validation = < X, Y, S, δint, δext, λ, ta >

Phase = {Passive, Active}

S = {Phase, sigma}

X = {“Control In”, integer}

δext (S,e,X)

{

Control\_In:

Case Phase {

Passive:

Sigma = Processing\_Time;

Phase = Active

}

Active:

//Ignore, should not happen

}

( λ) (S) {

Case Phase {

Active:

Send (Control\_Out\_port, ‘1’);

Passive:

// Do nothing

}

}

δint (S) {

Case phase {

Active:

Phase = passive;

sigma = infinity;

Passive:

// Ignore

}

}

**Coupled Model:**

**Financial Manager**: A sub model which gets the call and then checks if the customer has a valid coverage plan, if plan is valid the go out , if not credit card verification is required before valid plan gets rechecked again. If the plan is invalid and credit card verification fails as well the call will end abnormally.

Validation

Coverage Plan Manager

Problem

Financial Manager

Status

CC Info

Call in

Release Next

Plan OK

Donen

Input Ports: CALL\_INOutput Ports: PROBLEM, DONE

Financial Manager Formal Specifications:

Financial Manager= <X, Y, D, EIC, EOC, IC, SELECT>

X = <Call\_in>

Y = <Done, Problem>

D = <Coverage\_Plan\_Manager, Validation>

EIC = {Financial\_Manager.Call\_in, Coverage\_Plan\_Manager.Call\_in}

EOC = {(Coverage\_Plan\_Manager.Plan\_Ok, Financial\_Manager.Done),

(Coverage\_Plan\_Manager.Problem, Financial\_Manager.Problem)}

IC = {(Coverage\_Plan\_Manager.Request\_Validation, Validation.Request\_Validation),

(Validation.Result, Coverage\_Plan\_Manager.Validation\_Result)}

SELECT : {Coverage\_Plan\_Manager, Validation} = Coverage\_Plan\_Manager

**Atomic Models:**

1. **Coverage Plan Manager**: Which checks for valid coverage plan and if it is Ok sends value 1 through port Plan\_Ok .If the coverage plan is not valid, credit card validation is required there for validation request gets sent through port Request\_Validation. The result of result of validation will be sent back to the Coverage Plan Manager via port Validation\_Result. If the result of validation was Not Ok, value 1 gets sent through port Problem.

Plan Ok

Coverage Plan Manager

Problem

Call In

Validation Result

Request Validation

Input Port: Call\_In, Validation\_Result

Output Port: Plan\_Ok, Request\_Validation, Problem

Formal Specification for Coverage Plan Manager = < X, Y, S, δint, δext, λ, ta >

X = {(“Call\_Id”, integer), (“Validation\_Result”, {1,0})}

Y = {(“Plan\_Ok”, integer) , (”Problem”,integer), (“Request\_Validation”, integer)}

S = {Phase, sigma, Call\_In, Validation\_Result}

δext (S,e,X)

{

Case Phase {

Passive:

Case Port {

Call\_In:

Sigma = Processing\_Time;

Phase = busy;

RandNo = Genereate random number;

If (RandNo<=0.5) {

// Assuming half have valid plan and half don’t or they are new customers

Plan\_Valid = 1;

} else {

Need\_Validation = 1;

}

Validation Request:

Sigma = Processing\_Time;

Phase = busy;

Result = Validation\_Result.value

If (Result == 1) {

// Assuming credit card info were valid

Valid\_CC = 1;

} else {

Valid\_CC = 0;

}

}

busy:

//Ignore

}

}

( λ) (S) {

Case Phase {

busy:

If(Call\_In port){

If (Plan\_Valid == 1){

Send (Plan\_Ok\_port, Call\_In);

} elsif (Need\_Validation == 1){

Send (Request\_Validation\_port, Request\_Validation);

}

If (Validation\_Result\_port) {

If (Valid\_CC) {

Send (Plan\_Ok\_port, Call\_In);

} else {

Send(Problem\_port, Call\_In);

}

} else{

Passive:

// Ignore, should not happen

}

}

}

δint (S) {

Case phase {

busy:

phase = passive;

sigma = infinity;

passive:

// Ignore

}

}

1. **Validation:** If there was no plan for the calling customer of if the plan was expired, the validation will check if the credit card provided is valid. If valid new coverage plan is created and the call gets sent to the IT Help Desk Q, and a message gets sent to the financial queue to release next waiting call. If the credit card validation fails, the caller has to check the problem, and disconnect and release next signal is passed to the financial queue to generate next signal.

Result

Request Validation

Validation

Input Port: Request\_Validation

Output Port: Result

Formal Specification for Validation = < X, Y, S, δint, δext, λ, ta >

X = {“Request\_Verification”, Call Id}

Y = {(“Status”, {1,0}) }

S = {Phase, sigma, Valid, Invalid}

δext (S,e,X)

{

Case Phase {

Passive:

Sigma = Processing\_Time;

Phase = busy;

RandNo = Genereate random number;

If (RandNo<=0.8) {

// Assuming 80% will validate and only 20% will fail validation

Validate = 1;

} else {

Validate = 0;

}

busy:

//Ignore

}

}

( λ) (S) {

Case Phase {

busy:

Send (Result\_port, Status);

Passive:

// Ignore, should not happen

}

}

δint (S) {

Case phase {

busy:

phase = passive;

sigma = infinity;

passive:

// Ignore

}

}

**Testing Strategies:**

**Part A) Atomic Models**

* **Financial Q**

We add different incoming calls to the queue and watch the role of release next msg in releasing waiting calls in the Financial Q out of the queue through call out port, in the output file.

00:30:00:00 Call\_In 777

00:30:00:00 Call\_In 590

00:32:00:00 Call\_In 800

00:35:00:00 Release\_Next 1

00:40:00:00 Call\_In 444

00:45:00:00 Release\_Next 1

00:46:00:00 Release\_Next 1

The out put file is:

00:35:03:975 fq\_out 777

00:45:01:592 fq\_out 590

00:46:01:915 fq\_out 800

* **Financial Q Control**

Testing is done by sending inputs through control in port which should generate control out through control out port.

The input is:

00:30:00:000 Control\_In 434

The out put files will have:

00:00:03:975 control\_out 1

00:30:01:592 control\_out 1

First control out is enabling the Financial Q and required to get the first call out of the queue. Second control out output corresponds to the the Control\_In port. Regardless of previous results either if it came from Problem port or Plan\_Ok port of the Financial Manager coupled model.

* **Coverage Plan Manager**

This model is responsible for checking if the caller has valid coverage plan, if yes it will send the call through Plan\_Ok output port otherwise will send Request\_Validation message to the Validation model via Request\_Validation port.

The inputs are:

00:15:10:00 Call\_In 150

00:20:15:00 Call\_In 234

00:26:17:00 Call\_In 654

Which will generate the following output:

00:15:13:975 request\_validation 150

00:20:16:592 plan\_ok 234

00:26:18:915 plan\_ok 654

Based on our assumption, 50% of the calls already have valid coverage plan and 50% of them don’t either new customers or old customers with expired coverage plan.

* **Validation**

This model with gets validation request through its input port and then returns either 1 or 0 respectively success or failure to the coverage plan manager. The validation processing time is fixed and it takes 3 minutes.

The testing was done for the following inputs:

00:10:00:00 Request\_Validation 343

00:17:00:00 Request\_Validation 500

00:22:00:00 Request\_Validation 600

And the output file is:

00:13:00:000 result 1

00:20:00:000 result 1

00:25:00:000 result 1

**Part B) Coupled Models, Top Level and Second Level**

* **Financial Manager**

In the input file we have the following:

00:10:10:00 Call\_In 150

00:15:25:00 Call\_In 234

00:20:37:00 Call\_In 654

The output file is:

00:10:13:975 plan\_ok 150

00:15:26:592 plan\_ok 234

00:20:38:915 plan\_ok 654

Since none of the calls felt into validation request path, i.e. all had valid plan, for testing purposes we will change the possibility of having valid plan to %40 so that %60 of callers will need validation and at the same time will change the possibility of successful validation to %50 percent and will see the result having same input.

The output file is:

00:13:15:567 problem 150

00:18:30:985 plan\_ok 234

00:20:39:365 plan\_ok 654

* **Technical Support Financial Dept**

As mentioned before the code is assuming that only 40% of caller have valid plan and the rest need to get a new plan and validate their credit card. The credit card validation is 50% successful and 50% failed. Now to run the overall model we will have the following inputs:

00:10:00:00 Call\_In 430

00:15:00:00 Call\_In 234

00:20:00:00 Call\_In 777

00:25:00:00 Call\_In 998

00:30:00:00 Call\_In 223

00:50:00:00 Call\_In 900

00:59:00:00 Call\_In 111

And the output generated is:

00:13:07:577 problem 430

00:18:15:330 done 234

00:23:11:836 problem 777

00:28:18:234 done 998

00:30:04:474 done 223

00:50:12:189 done 900

00:59:10:010 done 111