# Modeling Discrete Event System Using DEVS

# (2014 Fall)

Assignment 1 – Simulation of Airplane take off procedure

Student name: Zheng Xia

Student number: 7356005

University of Ottawa

## PartⅠ

1. Problem Statement

When a plane comes to the bridge and before it actually takes off, there are numerous trivial yet vital steps need to be processed. It needs a collaboration from the customers, flight crews and the tower, sometimes the ground crews will be added in some other sophisticated systems as well. Therefore, a simulation for the whole procedure is necessary. The goal of building this simulator is to simulate the taking off procedure of an airplane. That is, those who wants to enhance their ability of airport management by using this simulator, could understand better on the collaboration of the different department.

1. Model Structure

Figure 1. Structure of Airplane take off procedure

As mentioned above, the take-off procedure contains three models: passenger queue, flight crew and tower. A general scenario can be describe as follows:

* 1. Passenger queue

This model is used to make all the passengers starting a line, and be processed in order and distribute those passengers wait the cabin ready (international passengers) to the cabin.

* 1. Flight

This model describes the preparation from the captain and service inside the plane as well as the luggage.

* 1. Tower

Tower module simulate a serial commands that sent from the air traffic controller, which only concerns the main steps of the taxiway activity.

1. Component Behavior
   1. Buffer: This atomic model aims at making all the passengers waiting for handling procedures in a queue. Once it receives any notification from the proc module, it will let the next passenger go ahead.
   2. Proc: the main function of this model is that it distributes passengers to the right seat. Yet, it does not do anything at the beginning except waiting the signal from the “cabin prepare”. It will be activated as soon as it receives the right output.
   3. Luggage: this module is used to deal with the passengers’ luggage.
   4. Cabin prepare: the module is being used as a pre-check of the passenger compartment and it will sent the “finish” signal to the proc so that the boarding procedure can launch.
   5. Flight plan: It would simulate some actions done by the captain.
   6. State check: It would read the state from the proc, flight plan and luggage then send the signal to tower controller.
   7. Clearance: It would check if the taxiway is busy and send message.
   8. Trailer truck: It would push back the airplane to the taxiway
   9. Sliding: It would decide whether the plane can take off or not.

## PartⅡ

As mentioned on figure 1, there are 1 input and 3 outputs in the take-off procedure. The *PassengerIn* means the numbers of passengers that will go aboard (in this case, the passengers are in the same airplane). All three outputs happens in the tower model, showing the disparity reasons of unsatisfied take-off. That is, *stateCheckFail* represents unsafety of plane cabin; *taxiwayBusy* means taxiway is being occupied and *skyBusfirst* describe the failure of flow control. Generally, the airplane take-off procedure model is made up of 3 components: Passenger Queue model, Flight model and Tower model. In addition, the Passenger Queue model can be decomposed into two components: buffer and proc, which simulates the lineup procedure of passengers and the time of boarding procedure, respectively. Furthermore, the Flight model is from the sight of flight crew, demonstrating the preparing inside the plane cabin and ground facility, which is consist of three components: luggage, cabin prepare and flight plan. They finish the part of the state checks collaboratively. The third is Tower model and there are 4 major components inside this couple model. These components check whether the procedure of the taking off is obeying the standard program.

### Formal specification of the atomic models

The more concrete DEVS formalism with port specifications is as follows:

DEVS\_model = < X, Y, S, δext, δint, λ, ta >

1. Buffer



X={bufferIn ,feedBackIn}

Y={bufferOut}

S={“InActive”,“Inpassive”,“OutActive”,“Outpassive”}

δint(“InActive”) = “InPassive”

δint(“OutActive”) = “Outpassive”

δext(“InPassive”,”buffIn) = “InActive”

δext(“OutPassive”,feedBackIn) = “OutActive”

λ(“InActive”, bufferTime) = 1 if outport = preActOut

= 0 if outport = buffOut

λ(“OutActive”, bufferTime) = 1

ta(“InActive”, bufferTime) = 5

ta(“OutActive”, bufferTime) = 5

ta(“InPassive”) = ∞

ta(“OutPassive”) = ∞

1. Proc



X={procIn}

Y={notificationOut, procOut}

S={procActive, procPassive, preparedActive, preparedPassive}

δint(“procActive”) = procPassive

δint(“preparedActive”) = preparedPassive

δext(“preparedPassive”, procIn) = “preparedActive”

δext(“procPassive”, procIn) = “procActive”

λ(“preparedActive”, procTime) = 1 if outport = cbStateOut

= 0 if outport = notificationOut

λ(“procActive”, procTime) = 1 if outport = procOut

= 0 if outport = notificationOut

ta(“procActive”, procTime) = 5

ta(“cabinActive”, procTime) = 5

ta(“procPassive”) = ∞

ta(“cabinPassive”) = ∞

1. Luggage



X={luggageIn}

Y={luggageReadyOut}

S={“luggageActive”, “luggagePassive”}

δint(“luggageActive”) = “luggagePassive”

δext(“luggagePassive”, luggageState) = luggageActive

λ(“luggageActive”, luggageTime) = 1

ta(“luggageActive”, luggageTime) = 5

1. Cabin prepare



X={cabinCrewIn}

Y={cabinCheckedOut}

S={“cabinActive”, “cabinPassive”}

δint(“cabinActive”) = “cabinPassive”

δext(“cabinPassive”, cabinState) = cabinActive

λ(“cabinActive”, cabinTime) = 1

ta(“cabinActive”, cabinTime) = 5

1. Flight plan



X={pilotIn}

Y={planeReadyOut}

S={“planActive”, “planPassive”}

δint(“planActive”) = “planPassive”

δext(“planPassive”, flightState) = planActive

λ(“planActive”, planTime) = 1

ta(“planActive”, planTime) = 5

1. State check



X={stateCIn}

Y={stateLugCOut, stateFligCOut, stateCOut, stateFail}

S={“stateLugCPassive”,”stateLugCActive”,”stateFligCPassive”,”stateFligCActive”,“stateCPassive”,”stateCActive”}

δint(“stateLugCActive”) = “stateLugCPassive”

δint(“stateFligCActive”) = “stateFligCPassive”

δint(“stateCActive”) = “stateCPassive”

δext(“stateLugCPassive”,stateCIn) = “stateLugCActive”

δext(“stateFligCPassive”,stateCIn) = “stateFligCActive”

δext(“stateCPassive”,stateCIn) = “stateCActive”

λ(“stateLugCActive”, procTime) = 1 if outport = stateLugCOut

= 0 if outport = stateFail

λ(“stateFligCActive”, procTime) = 1 if outport = stateFligCOut

= 0 if outport = stateFail

λ(“stateCActive”, procTime) = 1 if outport = stateCOut

= 0 if outport = stateFail

ta(“stateLugCActive”, procTime) = 5

ta(“stateFligCActive”, procTime) = 5

ta(“stateCActive”, procTime) = 5

ta(“stateLugCPassive”) = ∞

ta(“stateFligCPassive”) = ∞

ta(“stateCPassive”) = ∞

1. Clearance



X={clearIn}

Y={clearOut, clearFail}

S={“clearActive”, “clearPassive”}

δint(“clearActive”) = “clearPassive”

δext(“clearPassive”, clearState) = clearActive

λ(“clearActive”, clearTime) = 1 if outport = clearOut

= 0 if outport = clearFail

ta(“clearActive”, clearTime) = 5

1. Trailer truck



X={trailerIn}

Y={trailerOut}

S={“trailerActive”, “trailerPassive”}

δint(“trailerActive”) = “trailerPassive”

δext(“trailerPassive”, trailerState) = trailerActive

λ(“trailerActive”, trailerTime) = 1

ta(“trailerActive”, trailerTime) = 5

1. Sliding



X={slidingIn}

Y={slidingOut, flowFail}

S={“slidingActive”, “slidingPassive”}

δint(“slidingActive”) = “slidingPassive”

δext(“slidingPassive”, slidingState) = “slidingActive”

λ(“slidingActive”, slidingTime) = 1 if outport = slidingOut

= 0 if outport = flowFail

ta(“slidingActive”, slidingTime) = 5

### Formal specifications of coupled models

N= (X, Y, D, EIC, EOC, IC) for the coupled model Passenger Queue, Flight and Tower Simulator are defined as follows:

1. Passenger Queue

X = {bufferIn,cabinSCOut}

Y = {procStateIn}

D = {Buffer, Proc}

EIC = {((Passenger Queue,“bufferIn”),(Buffer,“buffIn”)),((Passenger Queue, “cabinSCOut”),(Proc, “cabinStateOut”))}

EOC = {(Proc, “procOut”) , (Passenger Queue, “procStateIn”)}

IC={((Buffer,“bufferOut”),(Proc,“procIn”)),((Proc,“notifyDone”),(Buffer,“procFeedback”))}

1. Flight

X = {FlightLuggage,FlightCabin,FlightPlan}

Y = {lugStateFin,planeStateFin,cabinSCOut}

D = {Luggage, Cabin Prepare, Flight Plan}

EIC = {((Flight, “FlightLuggage”), (Luggage, “luggageIn”)), ((Flight, “FlightCabin”), (Cabin Prepare, “cabinCrewIn”)), ((Flight, “FlightPlan”), (Flight Plan, “pilotIn”))}

EOC = {((Luggage, “luggageReadyOut”) , (Flight, “lugStateFin”)), ((Cabin Prepare, “cabinCheckedOut”) , (Flight, “cabinSCOut”)), ((Flight Plan, “planeReadyOut”) , (Flight, “planeStateFin”))}

IC = {}

1. Tower

X = {proStateIn, lugStateFin, planeStateFin}

Y = {stateFailOut, taxiwayBusyOut, flowControlOut, takeOffSucceed}

D = {State check, Clearance, Trailer truck, Sliding}

EIC = {((Tower, “procStateIn”), (State check, “stateCIn”)), ((Tower, “lugStateFin”), (state check, “stateCIn”)), ((Tower, “planeStateFin”),(state check, “stateCIn”))}

EOC = {((State check, “stateFail”) , (Tower, “stateFailOut”)), ((clearance, “clearFail”) , (Tower, “taxiwayBusyOut”)), ((Sliding, “flowFail”) , (Tower, “flowControlOut”)), ((Sliding, “slidingOut”) , (Tower, “takeOffOut”))}

IC = {((State check, “stateCOut”), (Clearance, “clearIn”)), ((Clearance, “clearOut”), (Trailer truck, “trailerIn”)), ((Trailer truck, “trailerOut”),(Sliding, “slidingIn”))}

### Test Strategies

Black box testing method is probably the best tools to test atomic models and coupled models in this case, which checks the internal transitions correctness. As for different atomic and coupled models, I will give different inputs, in order, to the corresponding event files (*.ev*), run each simulation and check whether outputs in output files (*.out*) are what I am expected to see.

As for this assignment, the sequence of testing should be following as:

1. “Luggage”, “Cabin Prepare” and “Flight Plan” atomic models
2. “Flight” coupled model
3. “Buffer” and “Proc” atomic models
4. “Passenger Queue” coupled model
5. “State check”, “Clearance”, “Trailer truck” and “Sliding” atomic models
6. “Tower” coupled model
7. “Airplane take off procedure” the top model

## PartⅢ

In order to ensure the correctness of the whole simulation, analyze the behavior of the atomic model is of importance. In this experiment, the atomic models will be presented followed by the couple model. The entire airplane take-off procedure model will be shown in the end.

1. Atomic model “Buffer”

The input of buffer should be a positive integer indicates whether the passenger arrives or not. By receiving the signal, the passenger can pass only after feedBackIn being received. from “Proc” ,which means the passenger should wait for the feedback at the port “feedBackIn”. Once it receives the feedback and then it could enter the next stage. The Buffer.ev I design as follows:

00:00:00:05 bufferIn 1

And the BufferOUT.out file in this case shows the expected as:

00:00:00:025 bufferout 1

1. Atomic model “Proc”

In this atomic model, the passenger comes from the “Buffer” should be put into a queue and if the queue is full, the model should inform the buffer to stop sending more people into the queue via another output port “notificationOut”. The whole procedure would be:

00:00:00:05 procIn 1

00:00:00:05 procIn 0

Which the expected .out file is:

00:00:00:025 procout 1

00:00:00:025 notificationout 0

1. Atomic model “Luggage”

Luggage mainly describes the state of the passenger’s luggage. It also could be used as a index for the state of the flight. The following shows the test event and the expected result:

00:00:00:05 luggageIn 1

And the result is:

00:00:00:025 luggageout 1

1. Atomic model “Cabin prepare”

There is another indication that can combining represent the flight state, which is the preparation of the flight crew. The experiment result and use event are given as follows:

00:00:00:05 cabinCrewIn 1

And

00:00:00:025 cabincheckedout 1

1. Atomic model “Flight plan”

The Flight plan model simulates the process of pilot’s preparation during the take-off procedure. The test and result are shown as follows:

00:00:00:05 pilotIn 0

And

00:00:00:025 planereadyout 1

1. Atomic model “State check”

In this model, the plane can be judged whether it is safety enough or not to going to the next stage that it should communicate with the tower controller for the discharging. The test input will be:

00:00:00:05 stateCIn 1

00:00:00:05 stateCIn 0

And the output represent if the plane can be pushed back:

00:00:00:025 statecout 1

00:00:00:025 statefail 0

1. Atomic model “Clearance”

This model simulates the communication between the pilot and the control tower. The experiment data and result should be:

00:00:00:05 clearIn 0

00:00:00:05 clearIn 1

And

00:00:00:025 clearfail 0

00:00:00:025 clearout 1

1. Atomic model ”Trailer truck”

Trailer truck describes the ground resources that could be used as an asset of discharging. The test on this model is:

00:00:00:05 trailerIn 1

And the result:

00:00:00:025 trailerout 1

1. Atomic model “Sliding”

This is the last step before actually taking off. Sometimes the sky will be busy and the plane will fail to take off due to the flow control. The test on this model is:

00:00:00:05 slidingIn 0

00:00:00:05 slidingIn 1

And the result is:

00:00:00:025 flowfail 0

00:00:00:025 slidingout 1

Respectively.

1. Couple model “Passenger Queue”

This couple model combine the functions on two separate atomic models: “Buffer” and “Proc” the link between these two are:

[top]

components : Buffer@Buffer Proc@Proc

in : bufferIn

out : procOut

Link : bufferIn bufferIn@Buffer

Link : bufferOut@Buffer procIn@Proc

Link : notificationOut@Proc bufferIn@Buffer

Link : procOut@Proc procOut

For the testing, the input event is:

00:00:00:05 bufferIn 1

The output is:

00:00:00:045 procout 1

1. Couple model “Flight”

This couple model combine the functions on three separate atomic models: “Cabin prepare”, “Flight plan” and “Luggage” the link between these models is:

[top]

components : Luggage@Luggage Cabin\_prepare@Cabin\_prepare Flight\_plan@Flight\_plan

in : luggageIn

in : cabinCrewIn

in : pilotIn

out : luggageOut

out : cabinCheckedOut

out : planeReadyOut

Link : luggageIn luggageIn@Luggage

Link : luggageOut@Luggage luggageOut

Link : cabinCrewIn cabinCrewIn@Cabin\_prepare

Link : cabinCheckedOut@Cabin\_prepare cabinCheckedOut

Link : pilotIn pilotIn@Flight\_plan

Link : planeReadyOut@Flight\_plan planeReadyOut

The test and the result is:

00:00:00:05 luggageIn 1

00:00:00:05 cabinCrewIn 1

00:00:00:05 pilotIn 1

00:00:00:025 luggageout 1

00:00:00:025 cabincheckedout 1

00:00:00:025 planereadyout 1

1. Couple model “Tower”

This couple model combine the functions on three separate atomic models: “Cabin prepare”, “Flight plan” and “Luggage” the link between these models is:

[top]

Components : State\_check@State\_check Clearance@Clearance Trailer\_truck@Trailer\_truck Sliding@Sliding

in : stateCIn

out : slidingOut

Link : stateCIn stateCIn@State\_check

Link : stateCOut@State\_check clearIn@Clearance

Link : clearOut@Clearance trailerIn@Trailer\_truck

Link : trailerOut@Trailer\_truck slidingIn@Sliding

Link : slidingOut@Sliding slidingOut

The test and the result is:

00:00:00:085 slidingout 1

1. The whole model – “Airplane take-off procedure”

The link between is shown as follows:

[top]

components : Buffer@Buffer Proc@Proc Luggage@Luggage Cabin\_prepare@Cabin\_prepare Flight\_plan@Flight\_plan State\_check@State\_check Clearance@Clearance Trailer\_truck@Trailer\_truck Sliding@Sliding

in : bufferIn

in : luggageIn

in : cabinCrewIn

in : pilotIn

%in : stateCIn

%out : procOut

%out : luggageOut

%out : cabinCheckedOut

%out : planeReadyOut

out : slidingOut

Link : bufferIn bufferIn@Buffer

Link : bufferOut@Buffer procIn@Proc

Link : notificationOut@Proc bufferIn@Buffer

%Link : procOut@Proc procOut

Link : luggageIn luggageIn@Luggage

%Link : luggageOut@Luggage luggageOut

Link : cabinCrewIn cabinCrewIn@Cabin\_prepare

%Link : cabinCheckedOut@Cabin\_prepare cabinCheckedOut

Link : pilotIn pilotIn@Flight\_plan

%Link : planeReadyOut@Flight\_plan planeReadyOut

Link : luggageOut@Luggage procIn@Proc

Link : procOut@Proc stateCIn@State\_check

Link : cabinCheckedOut@Cabin\_prepare stateCIn@State\_check

Link : planeReadyOut@Flight\_plan stateCIn@State\_check

%Link : stateCIn stateCIn@State\_check

Link : stateCOut@State\_check clearIn@Clearance

Link : clearOut@Clearance trailerIn@Trailer\_truck

Link : trailerOut@Trailer\_truck slidingIn@Sliding

Link : slidingOut@Sliding slidingOut

The test used event is:

00:00:00:05 bufferIn 1

00:00:00:05 luggageIn 1

00:00:00:05 cabinCrewIn 1

00:00:00:05 pilotIn 1

The result is:

00:00:00:125 slidingout 1

### Conclusion

The Airplane take-off procedure Simulator model simulates the process of taking-off of an typical airplane. In addition, the hierarchical test for both atomic models and coupled models generate the expected results. The output data is reasonable and the time of output can be calculated according to the time set in each model. The execution results and the behavior of the model match the specifications of models. The Airplane take-off procedure model works exactly as expected according to the specifications.