SYSC 5104- Methodologies for Discrete-Event Modeling and Simulation - Fall 2004

Modeling Panama Canal Using DEVS Formalism

Rami T. M. Madhoun (100667489)

Panama Canal Model

Introduction

This model describes the behavior of Panama Canal using DEVS formalism. The canal consists of two groups of locks separated by a lake (Gatun Lake). Only one of the lock groups (Gatun Locks) is modeled here since the other group has the same behavior but works in a reverse order. The Gatun Locks consist of three locks. Lock1 is lower than lock2 and lock3 is higher than both. So, in order for the ship to pass through, the water level of lock1 has to be lowered to the water level of the Atlantic Ocean so that the ship can move into lock1. The same process happens when the ship moves from lock1 to lock2, from lock2 to lock3 and then from lock3 to the Gatun Lake.

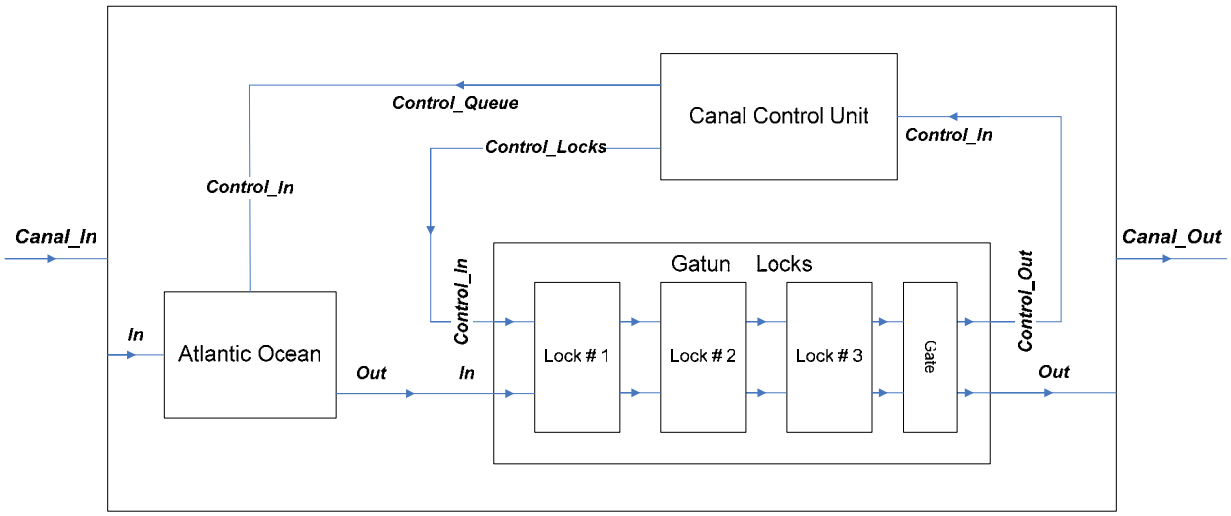


Figure 1

The main components of the model are as follows:

- **<u>The Atlantic Ocean (Queue)</u>**: This is where the ships start entering the canal and they are served on a first-come first-served basis. So, this component can be modeled as a queue.

- **<u>Canal Control Unit</u>**: This is the main control unit. It is responsible for performing the following:
  - Enabling/disabling the queue to regulate the ship traffic from the queue into the Gatun Locks.
  - Sending a control signal to the locks to receive a new ship.
  - Receives a signal from the locks whenever a ship leaves the lock successfully.

- **<u>Gatun Locks</u>**: is a coupled model composed of three locks and one gate. The locks operate as follows:
  When the control unit authorizes one ship to enter the locks, the following sequence of operations takes place:
  - Water flows from lock1 to the Atlantic Ocean until both reach the same water level, this process takes *wFlowTime* time units. Then, when the ship arrives at

lock1, the gate opens and the ship starts to move in. This process takes *transitionTime* time units.

- Water flows from lock2 into lock1 until both reach the same water level. This process take *wFlowTime* time units, after which, the gate of lock2 opens allowing the ship to move from lock1 to lock2.
- Water flows from lock3 into lock2 until both reach the same water level. This process take *wFlowTime* time units. Then, the gate of lock3 opens allowing the ship to move from lock2 to lock3.
- Water flows from the Gatun Lake through the Lock Gate until the water level in lock3 is the same as that in the Gatun Lake. After *wFlowTime* time units, the ship moves out and the locks send a signal to the control unit indicating that one ship has successfully passed through the canal.

Formal Definition of The Panama Canal Model

Coupled Models

Coupled Model Specification for the Panama Canal

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

X = {"Canal_In", integer}
Y = {"Canal_Out", integer}
D = {Queue, Gatun_Locks, Control_Unit}
EIC = {(Canal_In,Queue.In)}
EOC = {(Gatun_Locks.Out,Canal_Out)}
IC = {(Control_Unit.Control_Queue, Queue.ControlIn), (Control_Unit.Control_Locks, Gatun_Locks.ControlIn), (Queue.Out, Gatun_Locks.In), (Gatun_Locks.Control_Out,Control_Unit.ControlIn)}
SELECT: Priority order (descending) → Control_Unit, Queue, Gatun_Locks

Coupled Model Specification for Gatun Locks

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

X= {("In", integer) , ("Control_In", {1})}
Y = {("Out", integer), ("Control_Out",{1})}
D = {Lock, Lock_Gate}
EIC = { (In, Lock1.In), (ControlIn,Lock1.ControlIn)}
EOC = {(Gate.Out, Out), (Gate.Control_Out, Control_Out)}
IC = { (Lock1.Out, Lock2.In), (Lock1.Control_Out,Lock2.ControlIn), (Lock2.Out,Lock3.In), (Lock2.Control_Out, Lock3.ControlIn),(Lock3_Out, Gate_In),(Lock3.Control_Out, Gate.ControlIn)}
SELECT: Priority order (descending) → Gate, Lock3, Lock2, Lock1

Atomic Models

Atlantic Ocean (Queue)

The Atlantic Ocean (queue) works as follows:

- The queue is initially in a passive state (disabled).
- When ship arrives while the queue is disabled, it is added to the queue so that it can be served when its turn comes.
- The control unit sends two control signals, *Control_Queue* which activates the queue to send the first ship to the Gatun Locks. And *Control_Locks* which let the Gatun_Locks prepare for receiving the ship. This process takes a constant time *preparation_Time* after which the queue changes to passive state again.
- If ship arrives while another ship leaves the queue is also queued.
- If the ship arrives while no other ship in the queue → it will be queued waiting for the signal from control unit.

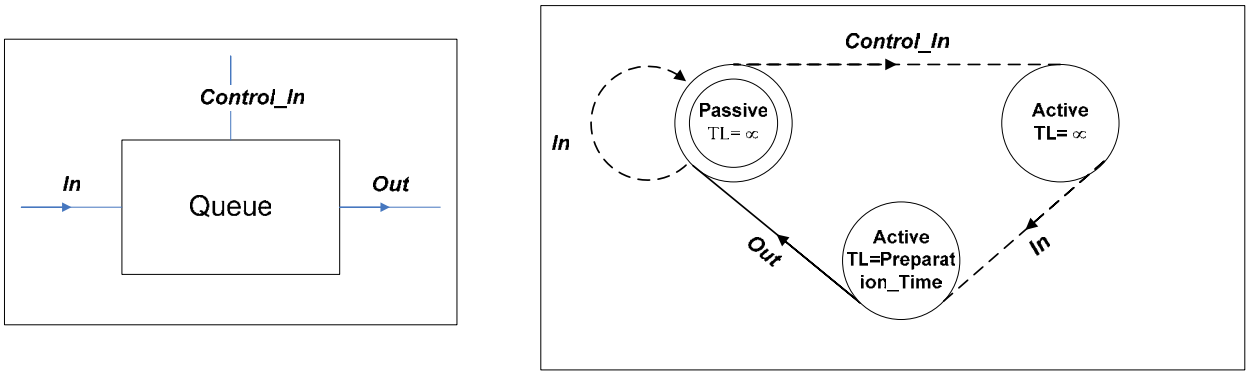


Figure 2

DEVS = < S , X , Y , δ_{int} , δ_{ext} , λ , ta >
Phase: {Active, Passive}
S = {Phase, σ, *No_of_Ships*}
X = { ("In", integer), (ControlIn, {1}) }
Y = {"Out", integer}

δ_{ext} (Phase, σ, *No_of_Ships*, e, x)
{
Case Phase {
Passive:
Case Port {

In:
No_of_Ships = *No_of_Ships* + 1;

ControlIn:

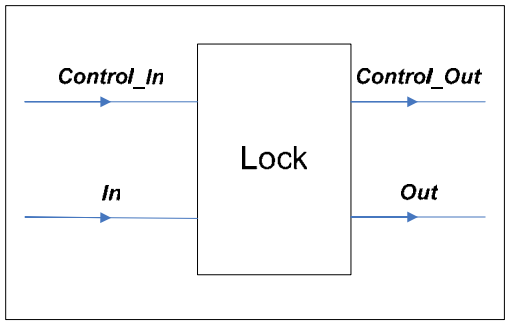
Phase = Active;
If *No_of_Ships = 0*

```
                    σ = ∞;
                else
                    σ = Preparation_Time;

        }
    Active:
        Case Port {

            In:
                If No_of_Ships = 0
                    σ = Preparation_Time;

                No_of_Ships = No_of_Ships + 1;

            Control In:
                This should not happen;    // the controller has to wait for the ship
                to pass through the locks before sending another controlIn signal.
        }
    }
}

δint (Phase, σ, No_of_Ships)
{
    Case Phase {
        Passive:
            This should not happen;
        Active:
            Phase = passive;
            No_of_Ships = No_of_Ships – 1;
            σ = ∞;
    }
}

λ(Phase, σ, No_of_Ships)
{
    Case Phase {
        Passive:
            Should not happen;
        Active:
            If (No_of_Ships > 0)
                Out = First Ship in Queue;
    }
}
```

Locks:

Locks are the atomic modes that constitute (with the Lock Gate) the Gatun Locks. They operate as follows:

- The lock will be initially, closed and water level will be high.
- When the lock receive a *ControlIn* signal, water will start to flow from the first lock into the Atlantic Ocean until the water level in both becomes the same (this process will take *wFlowTime* time units).
- At that point, a new ship will be ready to enter the lock (after the Queue *PreparationTime* has elapsed). This will open the lock gate and the ship will enter the Lock in *transitionTime* time units, then the lock door closes.
- Then, the lock will send a *Control_Out* Signal to the next lock so that water will start to flow from the next lock into the first one until they both reach the same level. This process takes *wFlowTime* time units.
- The lock will send the ship out through the output port which is connected to the input port of the next lock.
- The same process is repeated for each lock until lock3 is reached which will send a *Control_Out* signal to the lock gate to open and allow the water to flow from the Gatun Lake into Lock3 until their water level becomes the same. Then, the ship exists through the gate which will send *Control_Out* signal to the Canal Control Unit indicating that one ship has successfully passed through the locks.



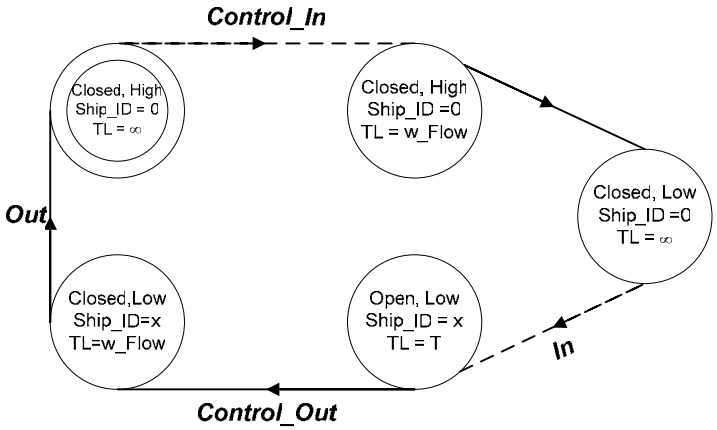


Figure 3

Phase: {{Open, Closed}, {Low, High} } // Low and High refer to the water level in the lock
S = {Phase, σ, *Ship_ID*}
X= {(In, integer),(ControlIn, {1}) }
Y = { (Out, integer), (Control_Out,{1}) }

δ_{ext} (S, e, x)
{
Case Phase {
Closed, High
Case Port{
ControlIn:
If *Ship_ID* = 0
Phase = Closed, High;
σ= *wFlowTime*;
else

```
                                        This should not happen;
                        In:
                                This shouldn't happen; // queue preparationTime ≥
                                                                wFlowTime
                }
        Closed, Low:
                Case Port{
                        ControlIn:
                                This should not happen;
                        In:
                                If Ship_ID = 0
                                        Phase = Open, Low;
                                        Ship_ID = In;
                                        σ = transitionTime;
                                else
                                        This shouldn't happen; // No two ships will be
                                        allowed to enter the locks at the same time;
                        }
        Open, High:
                        This can never happen; // water level can't be high if the door is
                                                open

        Open, Low:
                Case Port{
                        ControlIn:
                                This should not happen; // if the water level is already low,
                                no control signal should be sent
                        In:
                                This should not happens; // since the lock is open and the
                                water is low there must be a ship entering the lock, so no
                                other ship is allowed to enter at the same time
                }
        }

}

δint (Phase, σ, Ship_ID)
{
        Case Phase{
                Closed, High:
                        Phase = Closed, Low;
                        σ = ∞;

                Closed, Low:
                        If Ship_ID  != 0
                                Phase = Closed, High;
                                Ship_ID = 0;
                                σ = ∞;
```

Open, High:
This will never happen;
Open, Low:
If *Ship_ID* != 0
Phase = Closed, Low;
σ = *wFlowTime*
else
This shouldn't happen ; // The lock door will not open if there is no ship at the input port

}

}

λ(Phase, σ)

{

Case Phase {

Open, Low:
If *Ship_ID* != 0
Control_Out = 1;
Else
This shouldn't happen ;
Closed, Low:
If Ship_ID != 0
Out = *Ship_ID*;
Else
This should not happen;

}

}

Lock Gate:

Lock gate's function is to control water flow from the Gatun Lake into Lock3. Basically, After the ship enters lock3, lock3 will send *Control_Out* signal to the Lock Gate which will open to allow the water to flow into lock3 and the ship moves out. In addition, the lock gate will pass through the control signal to the Canal Control Unit, indicating that the ship has successfully passed the locks.

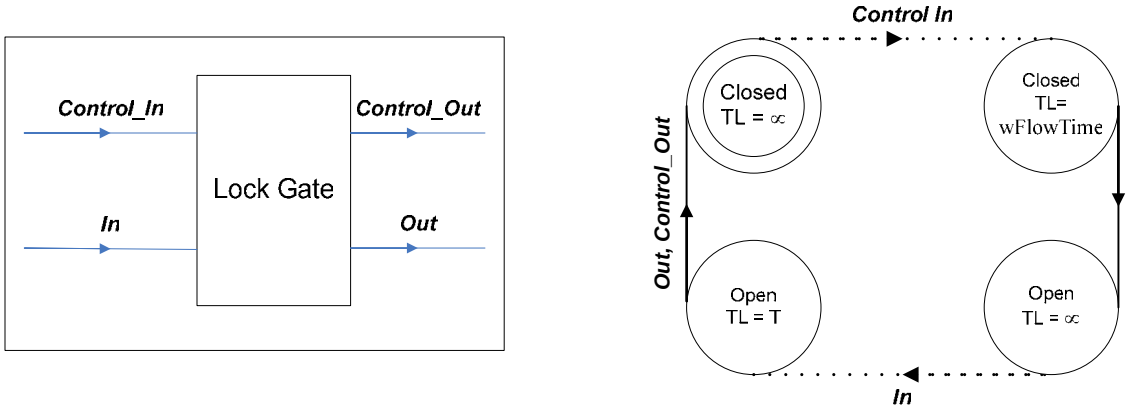


Figure 4

Phase = {Open, Closed}
S: {Phase, σ, *Ship_ID*}
X = {("In", integer),("ControlIn", {1})}
Y = {("Out", integer),("Control_Out", {1})}
δ_{ext} (S, e, x)

```
{
    Case Phase{
        Closed:
            Case Port{
                ControlIn:
                    Phase = closed;
                    σ = wFlowTime;
                In:
                    This should not happen;
            }
        Open:
            Case Port{
                ControlIn:
                    This should not happen;
                In:
                    Ship_ID = In;
                    σ = transitionTime ;
            }
    }
}
```

δ_{int} (S)

```
{
    Case Phase{
        Closed:
            Phase = Open;
            σ = ∞;
        Open:
            Phase = Closed;
            Ship_ID = 0;
            σ = ∞;
    }
}
```

λ(S)

```
{
    Case Phase{
        Closed:
            This should not happen;
        Open:
            If Ship_ID != 0
```

Out = *Ship_ID*;
Control_Out = 1;

}

}

Canal Control Unit:

This unit enables/disables the Queue , whenever the Gatun Locks become empty. In addition, it sends a *Control_Locks* signal to the Gatun Locks.

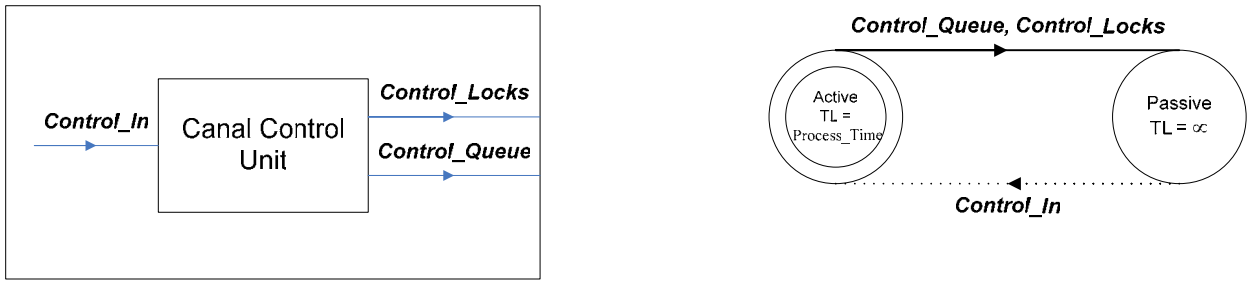


Figure 5

Phase = {Passive, Active}
S = {Phase, σ}
X = {("ControlIn",{1})}
Y = {("Control_Queue",{1}),("Control_Locks",{1})}

δ_{ext} (S, e, x)
{

Case Phase{

Active:

Case Port{

ControlIn:

This should not happen;

}

Passive:

Case Port{

ControlIn:

Phase = Active;
σ = *processingTime*;

}

}

}

δ_{int} (S)
{

Case Phase:

Active:

Phase = Passive;
σ = ∞;

Passive:

This should not happen;

}

}

λ(S)

{

Case Phase:

Active:

Control_Queue = 1;

Control_Locks = 1;

Passive:

This should not happen;

}

}

Model Testing

The testing strategy that will be followed depends on generating some testing scenarios for each atomic model separately and compare the actual output of the model with the expected one. After that, the coupled model is testing by injecting some input events and comparing the output of the coupled model with the expected output. This continues until we reach the full model hierarchy.

Atlantic Ocean (Queue):

The queue has two input ports (In, ControlIn) and one output port (Out). So different input will be injected to simulate ships arriving under different circumstances.
The *preparationTime* (time required by the ship to move to the gate of the Gatun Locks) is assumed to be **00:30:00:00 (h:m:s:ms)**. The *Ship_IDs* will be assigned numbers starting from 10,20,…etc

The input values are

Time	***Port***	***Value***	***Comments***
00:30:00:00	In	10	Will be queued until control signal arrives
00:35:00:00	ControlIn	1	Ship#10 will start to move out
00:45:00:00	In	20	Will be queued until control signal arrives
01:30:00:00	In	30	Will be queued until control signal arrives
02:00:00:00	ControlIn	1	Ship#20 will start to move out
04:00:00:00	ControlIn	1	Ship#30 will start to move out
05:00:00:00	In	40	Will start to move out immediately
06:00:00:00	In	50	Will not generate any output

The expected output is

Time	*Port*	*Value*
01:05:00:00	Out	10
02:30:00:00	Out	20
04:30:00:00	Out	30
05:30:00:00	Out	40

←→ queue.out

Comparing the expected output with the actual one in **queue.out** shows that the model behaves exactly as expected.

Locks:

The lock receives a control signal through *controlIn* port which will cause the water to flow from the lock until it reaches the same level as the previous lock (or the Atlantic Ocean). This happens in *wFlowTime* time units, after which the ship will be allowed to enter the lock. The ship will take *transitionTime* time units to enter the lock after which, the lock will send control signal to the next lock so that water will start to flow into the lock once the door closes. Then, water starts to flow in *wFlowTime* and then the model will generate an *Out* signal indicating that the ship is moving out of the lock.

The input values are

Time	*Port*	*Value*	*Comments*
00:30:00:00	controlIn	1	Starts water flow from the lock
00:35:00:00	controlIn	1	Invalidates the previous control signal
00:40:00:01	In	100	Will not generate any output
00:45:00:01	In	200	Will generate an output
02:00:00:00	controlIn	1	Prepares the lock to receive new ship
03:00:00:00	In	500	Will generate an output

The expected output is

Time	*Port*	*Value*
01:45:00:001	controlOut	1
01:55:00:001	Out	200
04:00:00:000	controlOut	1
04:10:00:000	Out	500

←→ lock.out

The actual output generated by **lock.out** matches the expected output which verifies the correctness of the model.

Lock Gate:

The lock gate receives a controlIn signal from lock3 in the Gatun Locks and opens to let the water flows from the Gatun Lake into Lock3. This will take *wFlowTime* time units, after which, the ship in lock3 will enter the gate and moves into the Gatun Lake, this process will take *transitionTime* time units. Then the lock will generate two output signals: *Out* which is the ship that passed through the gate and *controlOut* to inform the Canal Control Unit that the ship has passed through the Gatun Locks successfully.

transitionTime = **01:00:00:00 (h:m:s:ms)**

The input values are

Time	***Port***	***Value***	
00:20:00:00	In	15	Will not generate any output
00:30:00:00	controlIn	1	The gate will start to open
00:35:00:00	controlIn	1	This will invalidate the previous control Signal
00:45:00:01	In	25	Will generate an output
01:00:00:00	In	50	Will not generate any output

The expected output is

Time	**Port**	**Value**
01:45:00:001	out	25
01:45:00:001	controlOut	1

←→ gate.out

The actual output generated by **gate.out** matches the expected output which verifies the correctness of the model.

Canal Control Unit:

The canal control unit receives an input signal *controlIn* and generates two output signals *controlLocks, controlQueue* after the *processingTime* has elapsed.
The processingTime is assumed to be **00:05:00:000 (h:m:s:ms)**.
The input values are

Time	***Port***	***Value***	***Comments***
00:30:00:000	controlIn	1	Will generate an output
00:32:00:000	controlIn	1	Will not generate any output
00:40:00:000	controlIn	1	Will generate an output
01:00:00:000	controlIn	1	Will generate an output
01:04:59:000	controlIn	1	Will not generate any output

The expected output is

Time	*Port*	*Value*
00:05:00:000	controlLocks	1
00:05:00:000	controlQueue	1
00:35:00:000	controlLocks	1
00:35:00:000	controlQueue	1
00:45:00:000	controlLocks	1
00:45:00:000	controlQueue	1
01:05:00:000	controlLocks	1
01:05:00:000	controlQueue	1

←→ controller.out

Note: The first output was generated due to the fact that the Control Unit will be initiated with an active state.

Comparing the expected output with the actual one in **controller.out** shows that the model behaves exactly as expected.

Gatun Locks:

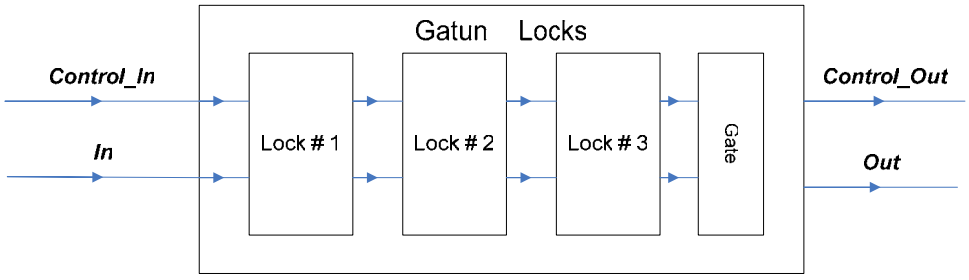


Figure 6

The Gatun Locks model is a coupled model consisting of three locks and one lock gate. When the ship enters the In port of the model:

- The ship will move from port In to Out in 3 * (*wFlowTime* + *transitionTime*) + transitionTime
- Another ship will be allowed to enter the locks after (*wFlowTime* + *transitionTime*) of the arrival of the first ship.

The input values are

Time	*Port*	*Value*	*Comment*
00:30:00:00	controlIn	1	
00:40:00:01	In	100	
01:50:00:02	controlIn	1	
02:00:00:03	In	500	
03:00:00:00	controlIn	1	
03:10:00:01	In	700	
03:10:00:04	controlIn	1	
03:20:00:05	In	1000	

The expected output is

Time	*Port*	*Value*
05:10:00:001	Out	100
05:10:00:001	controlOut	1
06:30:00:003	Out	500
06:30:00:003	controlOut	1
07:50:00:005	Out	1000
07:50:00:005	controlOut	1

←→ gatun_locks.out

The actual output generated by **gatun_locks.out** matches the expected output which verifies the correctness of the model.

Panama Canal:

The last step for testing the panama canal model is testing the complete coupled model (see Figure 1). The following points needs to be noticed when observing the input/output trajectories.

- Once the first ship arrives, the queue will be empty, the canal control unit will enable the queue and the Gatun Locks to receive the ship. This will take queue's preparationTime (*preparationTime* ≥ *processingTime*).
- The total time required for the ship to move from Canal_In port to the Canal_Out port can be calculated as follows:

Total Time = *preparationTime* + 3 * (*transitionTime* + *wFlowTime*) + *transitionTime*

- The minimum time period between any two consecutive ships leaving the canal is:

Minimum Time Period = **Total Time** + *processingTime*

preparationTime = **00:30:00:00**
transitionTime = **01:00:00:00**
wFlowTime = **00:10:00:00**
processingTime = **00:05:00:00**

The input values are

Time	*Port*	*Value*	*Comments*
00:00:00:00	In	300	Will generate an output
03:00:00:00	In	500	Will generate an output
06:00:00:00	In	600	Will generate an output
06:00:00:01	In	900	Will generate an output
10:00:00:00	In	1000	Will generate an output

The expected output is

Time	*Port*	*Value*
05:05:00:000	Out	300
10:10:00:000	Out	500
15:15:00:000	Out	600
20:20:00:000	Out	900
25:25:00:000	Out	1000

←→ panama_canal.out

The actual output generated by **panama_canal.out** matches the expected output which verifies the correctness of the model.