SAFETY ASSESSMENT
OF DYNAMIC SYSTEMS

Date: 06/02/2016

Developed by:

Sachin FERNANDES

Prerana SHAMSUNDAR PUNJABI

Table of Contents

Executive Summary

As an aircraft climbs, for the comfort of the passengers, the pressurization system will gradually increase the cabin altitude and the differential pressure at the same time. If the aircraft continues to climb once the maximum differential pressure is reached, the differential pressure will be maintained while the cabin altitude climbs. The maximum cruise altitude will be limited by the need to keep the cabin altitude at or below 8,000 ft.

A safety valve:

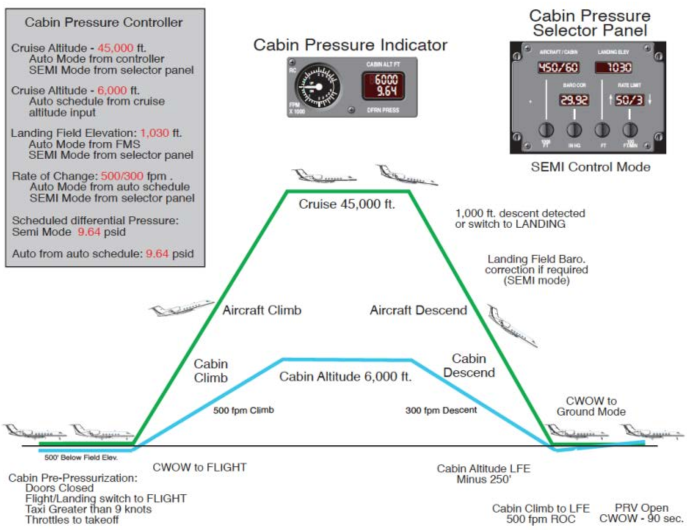
- acts as a relief valve, releasing air from the cabin to prevent the cabin pressure from exceeding the maximum differential pressure
- acts a vacuum relief valve, allowing air into the cabin when the ambient pressure exceeds the cabin pressure
- acts as a dump valve, allowing the crew to dump cabin air manually

A Cabin Altimeter, Differential Pressure Gauge, and Cabin Rate of Climb gauge help the crew to monitor the aircraft pressurization.

This assignment outlines problem statements were Failure Modes and Effects Analysis for the safety assessment of this dynamic system is to be developed. Failure Modes and Effects Analyses were completed for the system to ensure that the goals for the system had been meet.

The Dynamic System comprises mainly of the following:

- Three altitude levels
    - Ground (aircraft on ground)
    - Low (during landing and take-off)
    - High (cruise)
- Valves = Open when Aircraft altitude = Ground or Low [Cabin Altitude = Aircraft Altitude]
- Valves = Closed when Aircraft altitude = High [Cabin Altitude = Low]
- In normal Fight



1 Aircraft Altitude

<u>Design Description</u>: Aircraft Altitude Node

The Aircraft Altitude has got 3 states namely:

- `Ground`
- `Low`
- `High`

Events comprise of:

- `Take-off`
- `Up`
- `Down`
- `Land`

1.1 Code of Aircraft Altitude

In the below Trans: - The events `take_off`, `up`, `down` and `land` triggers a change in the Altitude of the aircraft if the respective conditions are met.

```
node Pressurization10_Pressurization10_Aircraft
 flow
  icone : [1,3] : out ;
  AAltitude : {ground,low,high} : out ;
 state
  Altitude : {ground,low,high} ;
 event
  take_off, up, down, land ;
 init
   Altitude := ground ;

trans
// transitions to be completed
Altitude = ground |- take_off -> Altitude := low;
Altitude = low |- up -> Altitude := high;
Altitude = high |- down -> Altitude := low;
Altitude = low |- land -> Altitude := ground;

assert
// assertions
AAltitude = Altitude ;

// Graphics
icone = case { Altitude= ground : 1,
               Altitude= low : 2,
               else 3 };
```

FIGURE 1: CODE FOR AIRCRAFT ALTITUDE

1.2 Simulation of Aircraft Altitude

The node was simulated successfully.

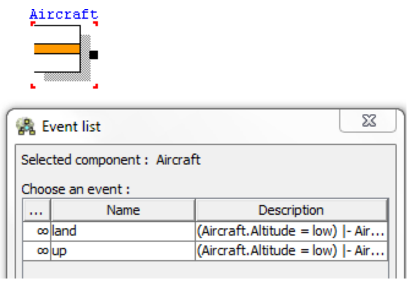


FIGURE 2: EVENT LIST OF THE NODE

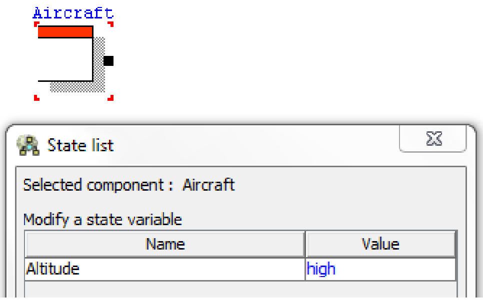


FIGURE 3: STATE OF THE NODE

The above simulation shows that the respective states changes happen when the events are triggered.

2 Cabin Altitude

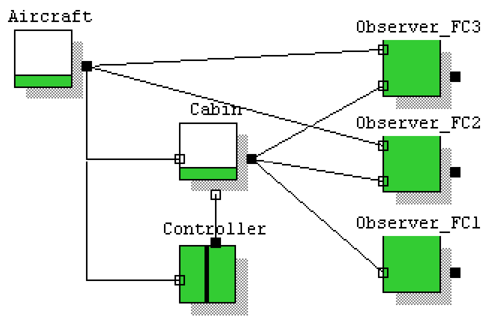


FIGURE 4: CABIN ALTITUDE CONTROLLER

2.1 The system with faults

<u>Design Description</u>: Cabin pressure is determined by considering the cabin altitude and the aircraft altitude. Incorrect variations in the cabin pressure might relate to faults in the system.

The faults that may arise are as described below:

2.1.1 Code for FC1:

`Cabin Altitude` is `High` implies that the cabin pressure is lower than desired pressure. This can cause lower level of oxygen in the aircraft and can cause death of the passengers.

```
node Pressurization10_Pressurization10_Observer_FC1
 flow
  icone : [1,2] : out ;
  O : bool : out ;
  CAltitude : {ground,low,high} : in ;

assert
// assertions

O =  (CAltitude=high)  ; // Failure Condition FC1 is true whenever the Cabin Altitude is high

icone = case { O=false : 1,
               else 2};
```

FIGURE 5: CODE FOR FC1

2.1.2 Code for FC2:

When the `aircraft` is in `cruise` and the `cabin altitude` is either `too low` (i.e. at ground level) or `too high.`

During cruise the cabin altitude is expected to be at 8000ft i.e. low level, anything value other than this can have major consequences

```
node Pressurization10_Pressurization10_Observer_FC2
 flow
  icone : [1,2] : out ;
  O : bool : out ;
  CAltitude : {ground,low,high} : in ;
  AAltitude : {ground,low,high} : in ;

assert
// Failure Condition FC2 is true whenever the Aircraft is in Cruise
// and the Cabin Altitude is different from what is required in a normal flight

O =  (AAltitude = high and (CAltitude = ground or CAltitude = high)) ;

// Graphics
icone = case {O=false : 1,
              else 2};
```

FIGURE 6: CODE FOR FC2

2.1.3 Code for FC3:

When the `aircraft` is in `ground` but the `cabin altitude` is still maintained at `high`

When the aircraft is on ground the cabin pressure should be more or less equal to the pressure at sea level, in case the pressure is more than this (i.e. either at low or high) this can cause again major consequences

```
node Pressurization10_Pressurization10_Observer_FC3
 flow
  icone : [1,2] : out ;
  O : bool : out ;
  CAltitude : {ground,low,high} : in ;
  AAltitude : {ground,low,high} : in ;

assert
// assertions
O =  (AAltitude = ground and (CAltitude = high or CAltitude = low)) ;
// Failure Condition FC3 is true whenever the Aircraft is on ground
// and the Cabin Altitude is different from what is required in a normal flight

// Graphics
icone = case {O=false : 1,
              else 2};
```

FIGURE 7: CODE FOR FC3

2.1.4 Minimal Cut Set without the Controller

The minimal cut set is defined as the minimal set of faults that will cause a failure on the desired application.

2.1.4.1 Minimal Cut set for FC1

```
/*
orders(MSS('Observer_FC1.O.true')) =
orders product-number
2      1
total  1
end
*/

products(MSS('Observer_FC1.O.true')) =
{'Aircraft.take_off', 'Aircraft.up'}
End
```

2.1.4.2 Minimal Cut set for FC2

```
/*
orders(MSS('Observer_FC2.O.true')) =
orders product-number
2      1
total  1
end
*/
products(MSS('Observer_FC2.O.true')) =
{'Aircraft.take_off', 'Aircraft.up'}
End
```

2.1.4.3 Minimal Cut set for FC3

```
products(MSS('Observer_FC3.O.true')) =
end
```

Events to trigger the fault: `Take_off -> up ->>> (altitude == high)` and `controller` is `false`->>>**Faults**. Cabin pressure is unaltered (remains at sea level)

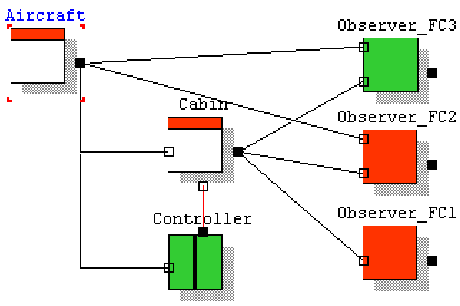


FIGURE 7: FAULT OCCURANCE DUE TO FAULTY CONTROLLER

2.2 The System with Controller

<u>Design Description</u>: Controller provides the commands to control the valve. `Close_cmd` shall be set to `true` in case the aircraft `altitude` is `high`.

<u>Note</u>: When the aircraft is in climbing stage, the cabin pressure gradually decreases. Since the rate of change of cabin altitude is lesser than that of aircraft altitude, the cabin takes time to get pressurized and maintain the sea level pressure at 8000 ft. The valve closes when this pressure is reached.

2.2.1 Code for the Controller:

```
node Pressurization10_Pressurization10_Controller
 flow
  icone : [1,2] : out ;
  AAltitude : {ground,low,high} : in ;
  close_cmd : bool : out ;

assert
// assertions
// Flawed Controller : the close_cmd is always true
close_cmd = (AAltitude=high) ;


// Graphics
icone = case { close_cmd=false : 1,
                         else 2};
```

FIGURE 8: CODE FOR CONTROLLER

2.2.1.1 Minimal Cut set for FC1 with the controller

```
products(MSS('Observer_FC1.O.true')) =
end
```

2.2.1.2 Minimal Cut set for FC2 with the controller

```
products(MSS('Observer_FC2.O.true')) =
end
```

2.2.1.3 Minimal Cut set for FC3 with the controller

```
products(MSS('Observer_FC3.O.true')) =
end
```

With the above controller design we can confirm that **`no failure conditions occur`**.

3 Faulty Transmitter

<u>Design Description</u>: Transmitter node provides a link between the controller and pressure control valve.

The transmission line alters its status to `closed` or `open` when events `stuck_open` and `stuck_close` occur.

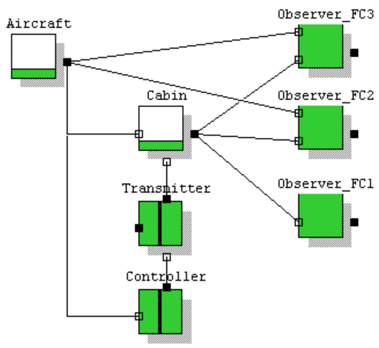


FIGURE 9: TRANSMITTER INCLUDED

3.1 Code for Transmitter

```
node Pressurization10_Pressurization10_FaultyTransmit
 flow
  icone : [1,4] : out ;
  close_cmd_O : bool : out ;
  close_cmd_I : bool : in ;
  Status_O : {ok,closed,open} : out ;
 state
  Status : {ok,closed,open} ;
 event
  stuck_closed, stuck_open ;
 init
   Status := ok ;

trans
//transitions
// When it is working correctly, the transmission line can be stuck closed
(Status = ok)|- stuck_closed -> Status:=closed;
// When it is working correctly, the transmission line can be stuck open
(Status = ok) |- stuck_open -> Status:=open ;
assert
// assertions
close_cmd_O = case {Status=ok :close_cmd_I ,
               Status=closed :true,
               else false};
// Status_O is equal to the Status
Status_O = Status ;

// Graphics
icone = case { Status=ok and close_cmd_I=false : 1 ,
               Status=ok and close_cmd_I=true : 2 ,
               Status=open : 3,
               else 4};
```

FIGURE 10: CODE FOR TRANSMITTER

3.1.1 Minimal Cut set for FC1 when a faulty transmitter is introduced

```
/*
orders(MSS('Observer_FC1.O.true')) =
orders product-number
3      3
total  3
end
*/
products(MSS('Observer_FC1.O.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'Transmitter.stuck_open'}
{'Aircraft.take_off', 'Transmitter.stuck_open', 'Aircraft.up'}
{'Transmitter.stuck_open', 'Aircraft.take_off', 'Aircraft.up'}
End
```

3.1.2 Minimal Cut set for FC2 when a faulty transmitter is introduced

```
/*
orders(MSS('Observer_FC2.O.true')) =
orders product-number
3      4
total  4
end
*/
products(MSS('Observer_FC2.O.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'Transmitter.stuck_open'}
{'Aircraft.take_off', 'Transmitter.stuck_open', 'Aircraft.up'}
{'Transmitter.stuck_closed', 'Aircraft.take_off', 'Aircraft.up'}
{'Transmitter.stuck_open', 'Aircraft.take_off', 'Aircraft.up'}
End
```

3.1.3 Minimal Cut set for FC3 when a faulty transmitter is introduced

```
/*
orders(MSS('Observer_FC3.O.true')) =
orders product-number
3      1
total  1
end
*/

products(MSS('Observer_FC3.O.true')) =
{'Aircraft.take_off', 'Transmitter.stuck_closed', 'Aircraft.land'}
End
```

In this case as the **`output from the transmitter is not fed back to aircraft node`**. There is no way to compare the transmission line status. Hence there is an increase in faulty conditions that many cause the system to fail.

Events to trigger the fault: **case_1** – set `stuck_closed` while landing causes cabin pressure to be above sea level and **FC3**occurs

case_2 – set `Take_off ->stuck_open -> up ->>> (altitude == high & valve = open)`**->>>Faults**. Cabin pressure reduces drastically causing **FC2**&**FC1** to occur)

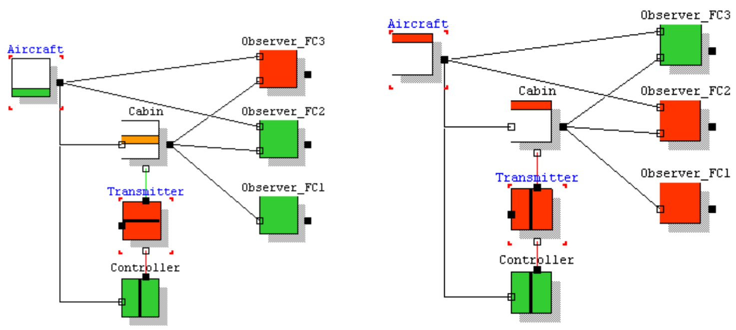


FIGURE 7: FAULT OCCURANCE DUE TO FAULTY TRANSMITTER

4 Failure Condition Avoidance

The status of the transmission link is provided as a feedback to the aircraft node. This helps in avoiding most errors that occur in the transmission.

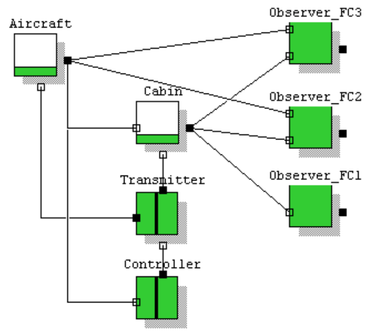


FIGURE 11: FAULT AVIODANCE

4.1 Code for the Aircraft Node with Transmitter link feedback

```
node Pressurization10_Pressurization10_Aircraft_feedback
 flow
  icone : [1,3] : out ;
  AAltitude : {ground,low,high} : out ;
  CStatus : {ok,closed,open} : in ;
 state
  Altitude : {ground,low,high} ;
 event
  take_off, up, down, land ;
 init
   Altitude := ground ;

trans
// transitions
Altitude = ground and (CStatus!=closed) |- take_off -> Altitude := low;
Altitude = low and (CStatus!=open) |- up -> Altitude := high;
Altitude = high  |- down -> Altitude := low;
Altitude = low   |- land -> Altitude := ground;

assert
// assertions
AAltitude = Altitude ;
// Graphics
icone = case { Altitude= ground : 1,
               Altitude= low : 2,
               else 3 };
```

FIGURE 12: CODE FOR AIRCRAFT NODE WITH TRANSMISSION LINE FEEDBACK

4.1.1 Minimal Cut set for FC1 with Transmitter feedback

```
/*
orders(MSS('Observer_FC1.0.true')) =
orders product-number
3      1
total  1
end
*/
products(MSS('Observer_FC1.0.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'Transmitter.stuck_open'}
End
```

4.1.2 Minimal Cut set for FC2 with Transmitter feedback

```
/*
orders(MSS('Observer_FC2.0.true')) =
orders product-number
3      1
total  1
end
*/
```

4.1.3 Minimal Cut set for FC3 with Transmitter feedback

```
products(MSS('Observer_FC2.O.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'Transmitter.stuck_open'}
End
products(MSS('Observer_FC3.O.true')) =
end
```

Events to trigger the fault:

case_1 – set `stuck_open` when the `aircraft` is `already in cruise` (Cabin pressure reduces drastically causing **FC2**&**FC1** to occur). This case cannot be avoided but the pilot can do damage control be landing as fast as possible.

case_2 – set `stuck_closed` when `aircraft is in ground` causes cabin pressure to be above sea level and **FC3** occurs

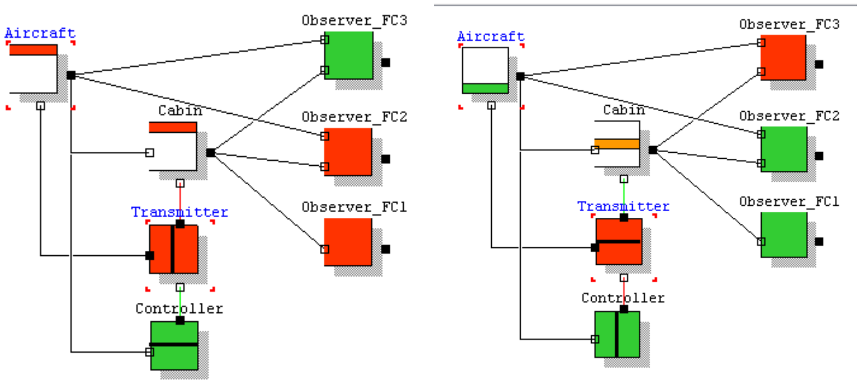


FIGURE 7: UNAVIODABLE FAULTS DUE TO MALFUNCTION – WITH TRANSMITTER FEEDBACK

Please note the faults scenarios that occurred before in model 3 without providing the feedback to the aircraft are avoided now. The above scenarios are special cases [**UNAVIODABLE malfunctioning devices**]. But at least the pilot can know these faults and try to take appropriate measure to handle the situation.

5 Failure Condition Tolerance

In this model a fault tolerance mechanism is used wherein 3 transmission links are utilized instead of 1. The controller checks for the maximum similar status and then computes the `close_cmd`

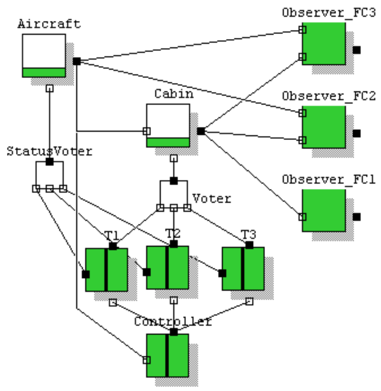


FIGURE 13: FAULT TOLERANCE

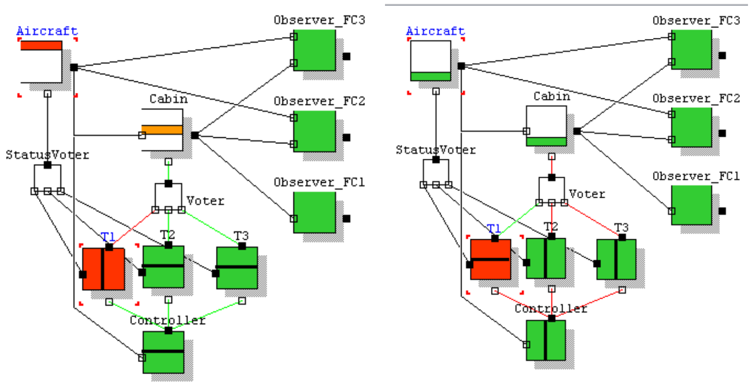


FIGURE 7: UNAVIODABLE FAULTS DUE TO MALFUNCTION – FAULT TOLERANCE MECHANISM

From the above simulation we observe that - if the same scenario that was stated in section 4 of malfunctioning devices occurs on this model the faults do not arise. This is because of the redundancy of the transmission lines. At least 2 out of 3 devices should fail and send faulty status for the failure conditions to occur. Which is more unlikely but can be possible? Hence this model is the best as we have reduced the risk of failure drastically.

5.1 Code for Voter

```
node Pressurization10_Pressurization10_Voter
 flow
  I1 : bool : in ;
  I2 : bool : in ;
  I3 : bool : in ;
  O : bool : out ;

assert
// assertions

//O = ((I1 and I2) or (I1 and I3) or (I2 and I3));
O = case {
        ((I1=I2) and (I2!=I3)): I1,
        ((I2=I3) and (I3!=I1)): I2,
        ((I3=I1) and (I1!=I2)): I3,
    else I1
};
```

FIGURE 14: CODE FOR VOTER

5.2 Code for Status Voter

```
node Pressurization10_Pressurization10_StatusVoter
 flow
  I1 : {ok,closed,open} : in ;
  I2 : {ok,closed,open} : in ;
  I3 : {ok,closed,open} : in ;
  O : {ok,closed,open} : out ;

assert
// assertions
O = case {
    ((I1=I2) and (I2!=I3)): I1,
    ((I2=I3) and (I3!=I1)): I2,
    ((I3=I1) and (I1!=I2)): I3,
    else I1
};
```

FIGURE 15: CODE FOR STATUS VOTER

5.2.1 Minimal Cut set for FC1 for fault tolerance

```
/*
orders(MSS('Observer_FC1.O.true')) =
orders product-number
4      18
total  18
end
```

```
*/
products(MSS('Observer_FC1.O.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open', 'T2.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open', 'T3.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open', 'T1.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open', 'T3.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open', 'T1.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open', 'T2.stuck_open'}
{'Aircraft.take_off', 'T1.stuck_open', 'Aircraft.up', 'T2.stuck_open'}
{'Aircraft.take_off', 'T1.stuck_open', 'Aircraft.up', 'T3.stuck_open'}
{'Aircraft.take_off', 'T2.stuck_open', 'Aircraft.up', 'T1.stuck_open'}
{'Aircraft.take_off', 'T2.stuck_open', 'Aircraft.up', 'T3.stuck_open'}
{'Aircraft.take_off', 'T3.stuck_open', 'Aircraft.up', 'T1.stuck_open'}
{'Aircraft.take_off', 'T3.stuck_open', 'Aircraft.up', 'T2.stuck_open'}
{'T1.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open'}
{'T1.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open'}
{'T2.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open'}
{'T2.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open'}
{'T3.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open'}
{'T3.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open'}
end
```

5.2.2 Minimal Cut set for FC2 for fault tolerance

```
/*
orders(MSS('Observer_FC2.O.true')) =
orders product-number
4      18
total  18
end
*/

products(MSS('Observer_FC2.O.true')) =
{'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open', 'T2.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open', 'T3.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open', 'T1.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open', 'T3.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open', 'T1.stuck_open'}
{'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open', 'T2.stuck_open'}
{'Aircraft.take_off', 'T1.stuck_open', 'Aircraft.up', 'T2.stuck_open'}
{'Aircraft.take_off', 'T1.stuck_open', 'Aircraft.up', 'T3.stuck_open'}
{'Aircraft.take_off', 'T2.stuck_open', 'Aircraft.up', 'T1.stuck_open'}
{'Aircraft.take_off', 'T2.stuck_open', 'Aircraft.up', 'T3.stuck_open'}
{'Aircraft.take_off', 'T3.stuck_open', 'Aircraft.up', 'T1.stuck_open'}
{'Aircraft.take_off', 'T3.stuck_open', 'Aircraft.up', 'T2.stuck_open'}
{'T1.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open'}
{'T1.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open'}
{'T2.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open'}
{'T2.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T3.stuck_open'}
{'T3.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T1.stuck_open'}
{'T3.stuck_open', 'Aircraft.take_off', 'Aircraft.up', 'T2.stuck_open'}
end
```

5.2.3 Minimal Cut set for FC3 for fault tolerance

```
products(MSS('Observer_FC3.O.true')) =
end
```

6 Conclusion

This project is a good opportunity to apply the advices and the lessons taught. The session gave us a unique exposure to realize a dynamic model and provided us with an insight on safety assessments of dynamic systems.