D7.5.1 – Library for generic ECS

WP 7.5: Extended Aircraft Environmental Control Systems

Work Package 6: Model component libraries

MODRIO (11004)

Version 0.1

Date 22/04/2013

Authors

Maxime Payelleville — Dassault Aviation

Summary

Executive summary

Two libraries were planned to be developed in MODRIO:

- A shared library for building a generic Environmental Control System (ECS) with no confidential information, and for defining standards of implementations.

- An internal library dedicated to simulation and support of state estimation of real vehicle management systems with some results from WPs 2.1, 3.1, 6.1, 6.2 and 7.5.1. It shall be usable in a real design process as well as appropriate for system health monitoring in operation.

This document describes the shared library improved during the project.

1. SUMMARY

This document describes the shared library (Modrio) improved during the project.

In MODRIO several work packages had been selected by Dassault Aviation to enhance the capabilities of Modelica and Dymola. Dependencies of WP 7.5 with other work-package are illustrated in the following figure:

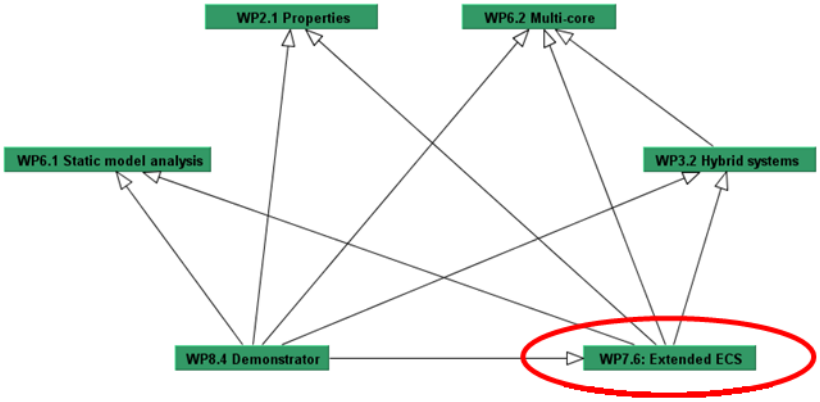


Figure 1 : WP dependencies

1.1 Terms

- APU : Auxiliary Power Unit
- ATA : Air Transport Association of America
- BAS : Bleed Air System
- BDD : SYSML Block Diagram Definition
- BIZJET : Business aircraft
- DMU : Aircraft Digital Mock-Up
- ECS : Environmental Control System
- EUROSYSLIB : Project ITEA2 EUROSYSLIB
- FMI/FMU : Functional Mock-up Interface / Unit
- IBD : SYSML Internal Block Diagram
- LRU: Line Replacable Units (items replaced by maintenance actions when failed)
- MSL: Modelica Standard Library (Short name: Modelica)
- MOE: MOre Electrical Aircraft
- TOICA : Project FP7 TOICA (Thermal Overall Integrated Concept of Aircraft)

2. MODRIO LIBRARY DESCRIPTION

2.1 Components of the shared libraries

A new library called Modrio has been developed to share and illustrate needs with Modrio partners.

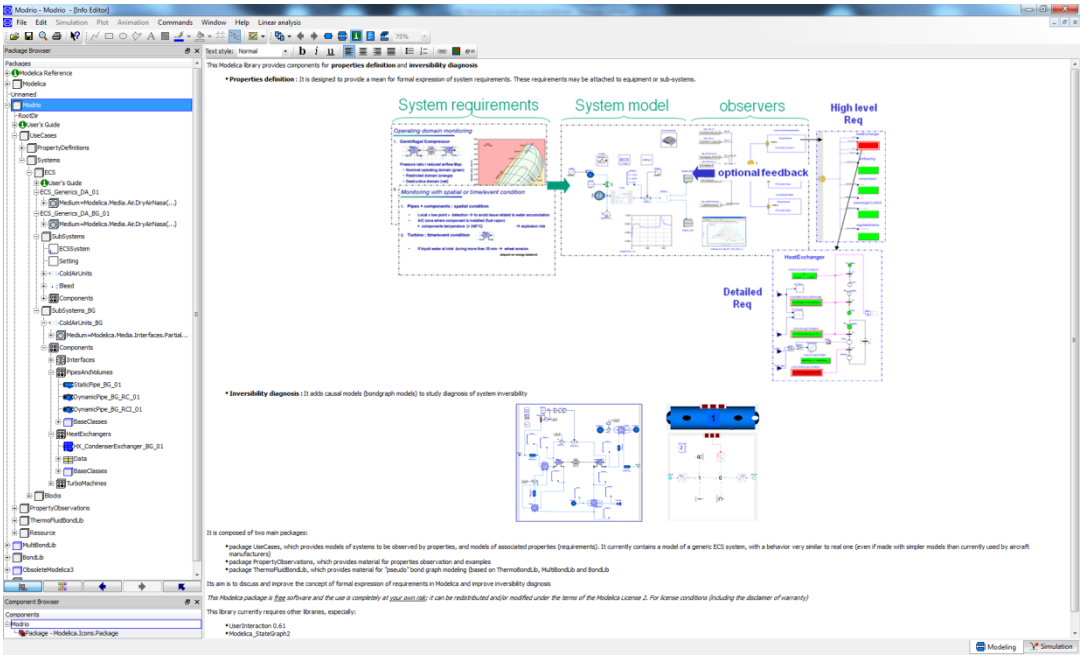


Figure 2: Dassault Aviation Modrio Library

It was created from the Properties library made during the ITEA2 project Eurosyslib for properties modeling investigations.

2.2 Additional components for structural analysis

The objective of the developed new features is to be compatible with current and associated models. These features must be integrated within other Aircraft Vehicle Systems models.

For the need of structural analysis WP6.1 ([R07]) use cases and bond graph components have been added to the library. The components are described in the following paragraph.

The purpose is to be able to change acausal components by causal bond graph components and allow the first steps of the invertibility analysis planed within WP6.1.

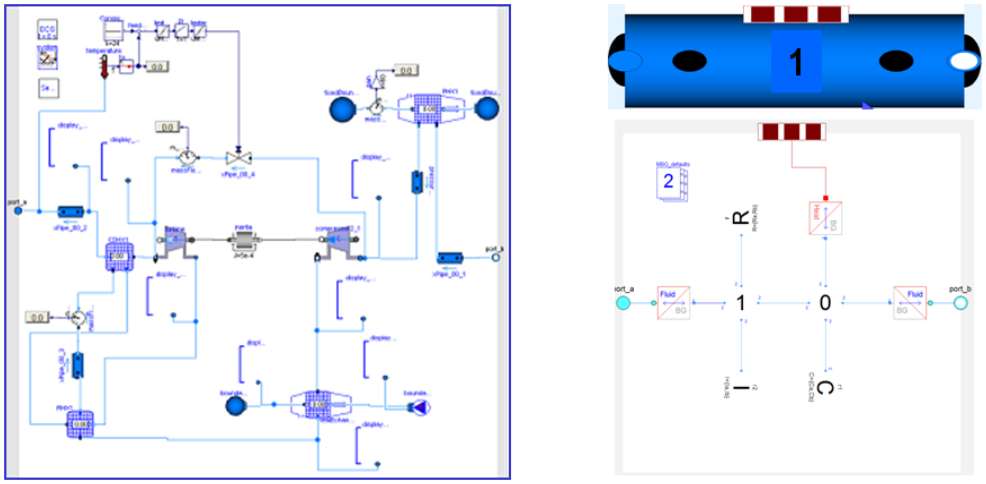


Figure 3: Dassault Aviation System

2.2.1 Models of components

2.2.1.1 Causal models

The only interface parts which change are the connectors. The purpose is to have the same parameters, but model described internally with bond graph components.

The following paragraph defines Modelica components with acausal ports made to be able to connect internally to bond graph ports of the MultiBonLib library, through a new package called ThermoFluidBondLib (with pseudo-bondgraph ports), which replace the library ThermoBondLib, too far from the need.

2.2.1.2 Assumptions

The following causal models are described hereafter only for dry air. For use with, moist air additional variables (ratio of water in air, liquid water amount …) and equations should be used in addition.

2.2.1.3 Connectors

The connector of the acausal model is the one of Modelica.Fluid:

```
connector FluidPort
  "Interface for quasi one-
dimensional fluid flow in a piping network (incompressible or compressible, one or more phas
es, one or more substances)"
```

```
  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model" annotation (choicesAllMatching=true);

  flow Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";
  Medium.AbsolutePressure p "Thermodynamic pressure in the connection point";
  stream Medium.SpecificEnthalpy h_outflow
    "Specific thermodynamic enthalpy close to the connection point if m_flow < 0";
  stream Medium.MassFraction Xi_outflow[Medium.nXi]
    "Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0";
  stream Medium.ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m close to the connection point if m_flow < 0";
end FluidPort;
```

The original connector of ThermoBondLib is made of pure bond graph variables:

```
connector ThBondCon "Bi-directional thermo-bond graph connector"
  Modelica.SIunits.Temperature T "Temperature";
  Modelica.SIunits.Pressure p "Pressure";
  Modelica.SIunits.SpecificEnthalpy g "Gibbs potential";
  Modelica.SIunits.ThermalConductance Sdot "Entropy flow";
  Modelica.SIunits.VolumeFlowRate q "Volume flow";
  Modelica.SIunits.MassFlowRate Mdot "Mass flow";
  Modelica.SIunits.Entropy S "Entropy";
  Modelica.SIunits.Volume V "Volume";
  Modelica.SIunits.Mass M "Mass";
  Real d "Directional variable";
  Boolean Exist "True if substance exists";

end ThBondCon;
```

Variables of this connector are very far from the Modelica.Fluid connector.

Therefore, a new connector, for "pseudo" bond graph has been created, to be closer to the acausal connector and allow fewer models modifications:

```
connector FluidPort
  "Interface for quasi one-
dimensional fluid flow in a piping network (incompressible or compressible, one or more phas
es, one or more substances)"

  import SI = Modelica.SIunits;

  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model" annotation (choicesAllMatching=true);

  Medium.AbsolutePressure p "Thermodynamic pressure in the connection point";

  Medium.Temperature T "Thermodynamic temperature in the connection point";
  //flow
  Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";

  //flow
  SI.Power H_inflow
    "Thermodynamic enthalpy mass flow rate close to the connection point if m_flow < 0";
```

```
end FluidPort;
```

MultibondLib bondgraph main connector:

```
connector MultiBondCon "bi-directional bondgraphic connector"
  parameter Integer n=1 "Cardinality of Bond connection";
  Real e[n] "Bondgraphic effort variable";
  //flow
  Real f[n] "Bondgraphic flow variable";
  Real d "Directional variable";

end MultiBondCon;
```

2.2.1.4 Interfaces between physical and Bond graph connectors

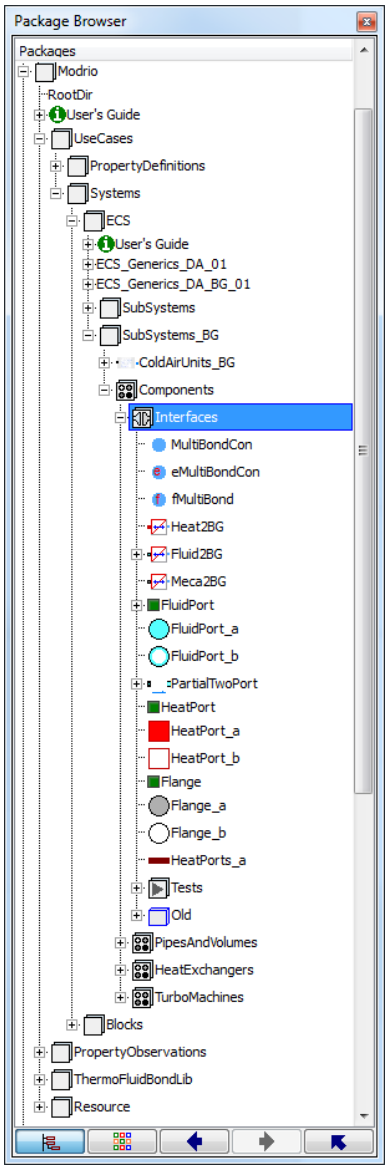
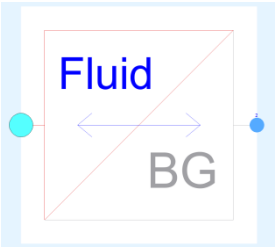


Figure 4: Interfaces and bond graph connectors

Components make the interface connections between internal bond graph connectors (from MultibondLib) and the "pseudo" bond graph connectors:

- Interface model between bond graph and fluid connector:

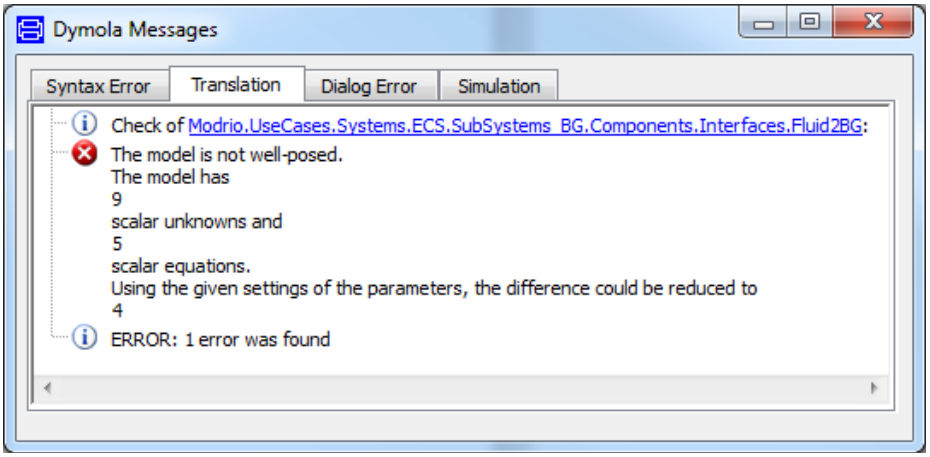


- Code:

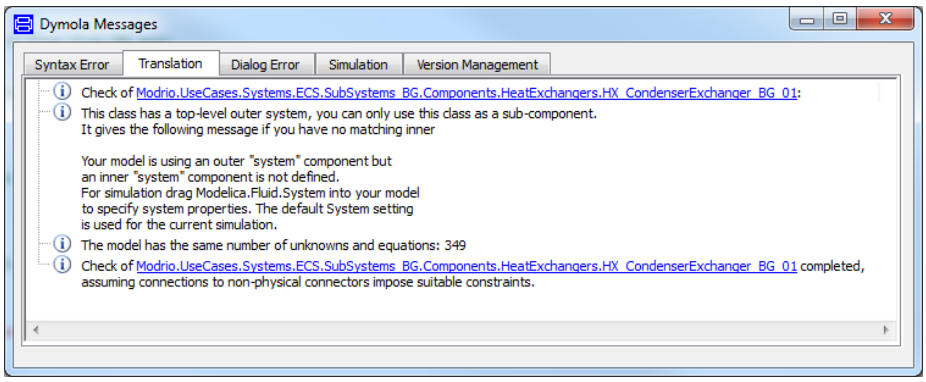
```
model Fluid2BG "acausal Fluid(P,T) to/from bond graph conversion"

 import SI = Modelica.SIunits;

 replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
  "Medium in the component" annotation (choicesAllMatching=true);

 Modrio.UseCases.Systems.ECS.SubSystems_BG.Components.Interfaces.MultiBondCon
  MultiBondCon1(final n=2) "Bond graph connector" annotation (Placement(
    transformation(extent={{90,-10},{110,10}}, rotation=0)));

 FluidPort_a port_a(redeclare package Medium = Medium)
  "Thermal fluid connector" annotation (Placement(transformation(extent={{-110,
      -10},{-90,10}}, rotation=0)));

 // parameter Integer n=1 "Cardinality of Bond connection";
 // Real e[n] "Bondgraphic effort variable";
 // Real f[n] "Bondgraphic flow variable";
 // Real d "Directional variable";

 // Medium.AbsolutePressure p
 // Medium.SpecificEnthalpy h
 // flow Medium.MassFlowRate m_flow
 // flow SI.Enthalpy H_inflow

equation
 MultiBondCon1.e[1] = port_a.p;
 MultiBondCon1.e[2] = port_a.T;
 MultiBondCon1.f[1] = -port_a.m_flow;
 MultiBondCon1.f[2] = -port_a.H_inflow;

 MultiBondCon1.d = -1;

end Fluid2BG;
```

- Current issues

  Components to transform physical variables to bond graph variables have physical connectors without type flow to be compatible with the MultibondLib library. Individually, they don't check because there are too many variables.



  Even if check of component which integrates previous interfaces is true, it is not completely correct. It is not satisfactory but it is not really important because the library is only developed for test.



2.2.1.5 Pressure loss

A pressure loss corresponds to the momentum balance equation. It gives relations between boundary variable (pressures, temperatures) and average mass flow rate going through the component.

- Causal boundary variables:

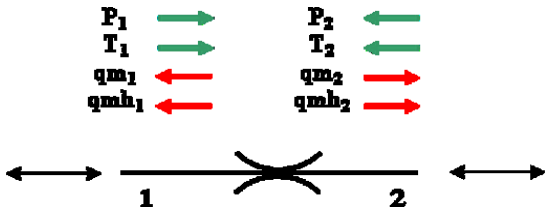


Figure 5: Pressure loss bond graph representation

- Equations:

For compressible media (like dry air), it is often given by the following equation:

$$qm_{12} = f(P_1, P_2, T_1, T_2)$$

The mass flow rate can be blocked if the flow is chocked (Mach = 1).

The pressure drop used with the components of the CAU is given by the following formula valid when Mach < 0.3:

$$\left|P_2^{\,2} - P_1^{\,2}\right| = 2.K.qm^{\alpha}.Tm$$

With $Tm = \frac{T_1 + T_2}{2}$ or $Tm = T_{up}$

And K is the pressure drop coefficient, which is different either the flow is laminar or turbulent. α is a constant coefficient close to 2.

- Bond graph representation:

  The bond graph representation of the pressure drop encapsulated in a Modelica component, also called Static Pipe, is represented by the following figure. The external ports of the component are Modelica connectors

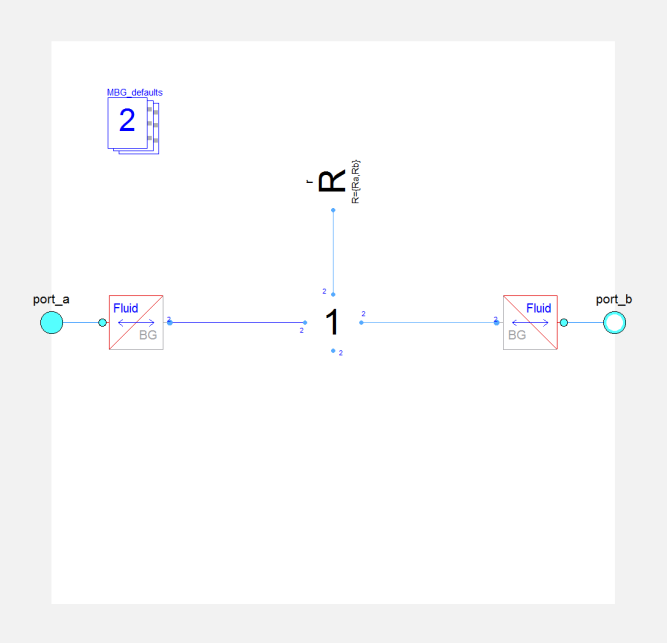


Figure 6: Static pipe bond graph representation

2.2.1.6 Volume

A volume is used to calculate the mass and energy balances. It gives static variables pressure (P) and temperature (T) within the volume.

- Causal boundary variables:

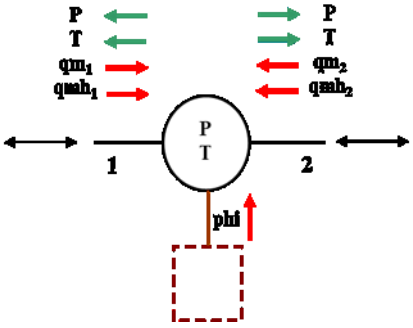


Figure 7: Volume bond graph representation

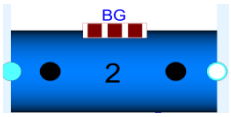
- Equations
  - Mass balance: $\frac{dm}{dt} = qm_2 + qm_i$

    m is the mass within the volume
  - Energy balance: $\frac{dU}{dt} = qmh_2 + qmh_i + \Phi$

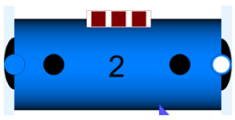
    U is the total internal energy within the volume

2.2.1.7 Dynamic pipes

- Icon of the causal model



- Icon of corresponding acausal model (similar to which of Modelica.Fluid):



According to the Finite Volume method, pipes can be divided into cells and each cell can have the behavior split into volume and resistance element. To fulfill the three balance equations of a cell, it is sufficient to describe the pipe with a volume and a pressure drop.

The simple pipe is composed with only one volume and one pressure drop, as defined above.

- Causal boundary variables, illustrated for a simple pipe:

- Element defined with a volume and a resistance (if we consider that the nominal flow is from port 1 to port 2):

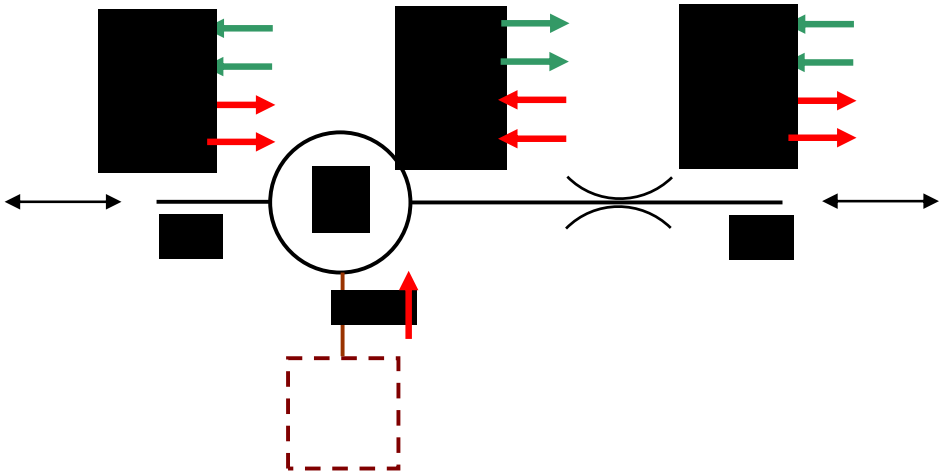


Figure 8: Dynamic pipe bond graph representation (volume + resistance)

- Element with a resistance and a volume resistance (if we consider that the nominal flow is from port 1 to port 2):

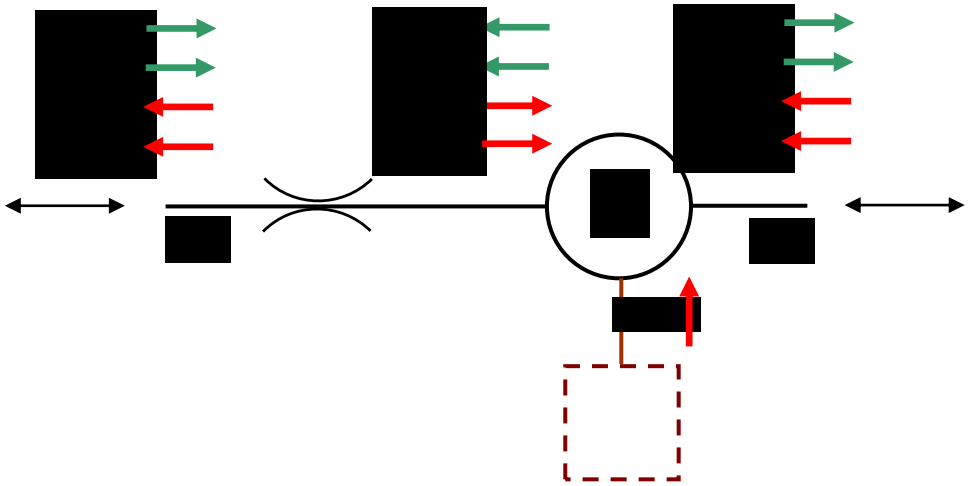


Figure 9: Dynamic pipe bond graph representation (volume resistance + resistance)

Remarks:

- ♦ Pipes may have reverse flow during a simulation.
- ♦ In an acausal world, these components would be the same. Connections and simulation ability depend of the internal structure of the component.

- Bond graph representation

  The bond graph representation of the pressure drop encapsulated in a Modelica component, also called Static Pipe, is represented by the following figure. The external ports of the component are Modelica connectors

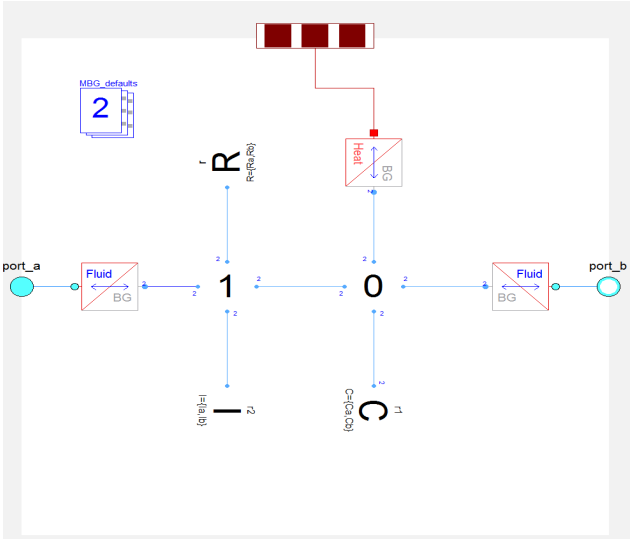
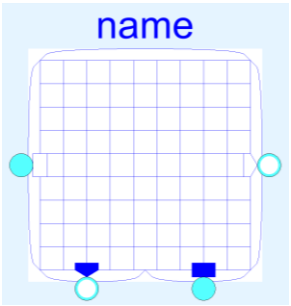


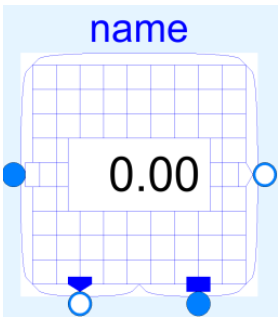
Figure 10: Pressure drop bond graph representation

2.2.1.8 Heat Exchangers

- Icon of the causal model (condenser heat exchanger):



- Icon of corresponding acausal model:



- Causal boundary variables, illustrated with a simple heat exchanger composed of simple pipes and a simple heat transfer:

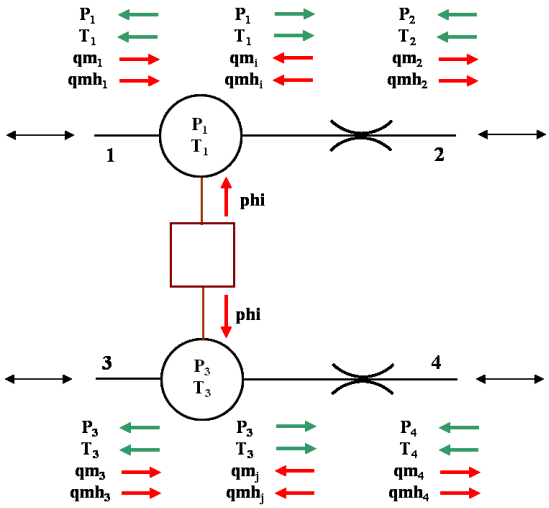
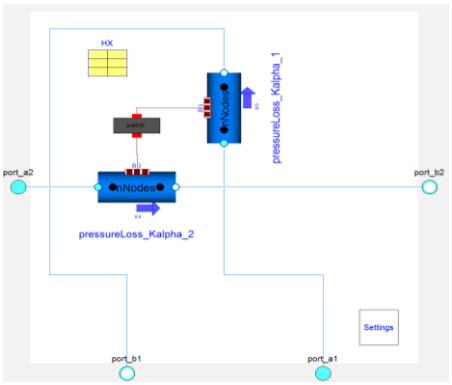


Figure 11: Heat exchanger bond graph representation

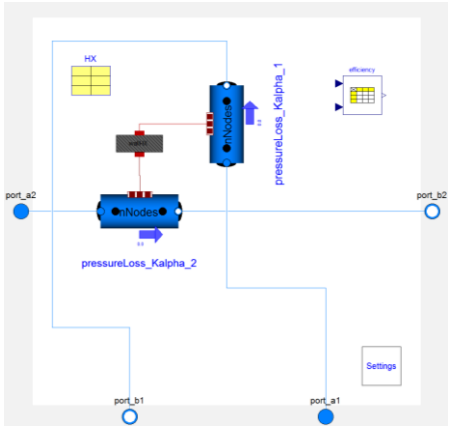
- Equations:

$$Tc_s - Tc_e = \varepsilon.(Tf_e - Tc_e)$$

- Internal composition of the causal model:



- Internal composition of the acausal model:

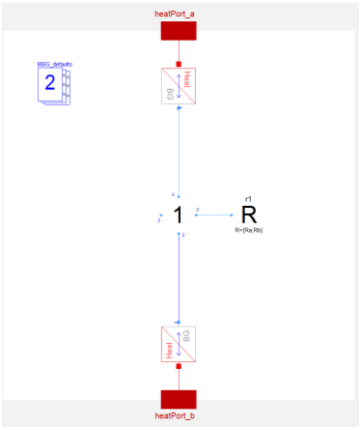


- Heat transfer component:

The bond graph representation of the heat transfer between pipes encapsulated in a Modelica component is represented in the following figure. The external ports of the component are Modelica connectors

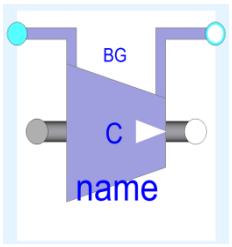
Icon:

- Internal composition:

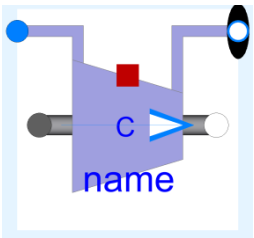


2.2.1.9 Compressor

- Icon of the causal model:



- Icon of the corresponding acausal model:



- Causal boundary variables for a simple compressor:

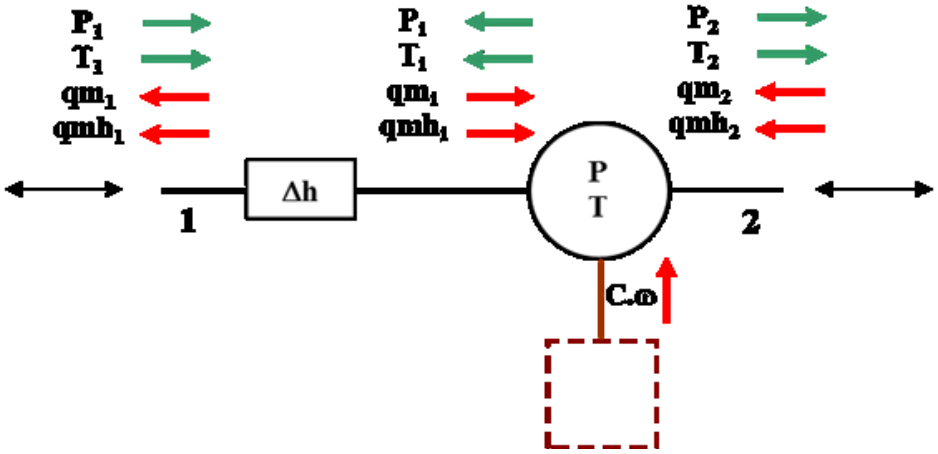


Figure 12: Compressor bond graph representation

- Equations:
  - ♦ Assumptions :
    - No internal leaks
    - No external heat transfer
  - ♦ Equations
    - Mass balance

$$\frac{dm}{dt} = qm_2 + qm_i$$

    - Energy balance

$$\frac{dU}{dt} = qmh_2 + qmh_i + C.\omega \qquad qmh_2 + qmh_i = -qm_1 \cdot \Delta h$$

$$\Delta h = f(\eta_{is}, h_1, P_1, T_1, P_2)$$

    - Mass flow rates

$$qm = qm_2 = f(\omega, P_1, T_1, P_2)$$

    - Efficiency (isentropic)

$$\eta = f(\omega, P_1, T_1, P_2)$$

    - Pressure ratio

$$\frac{P_2}{P_1} = f(qm, \omega, T_1, P_1)$$

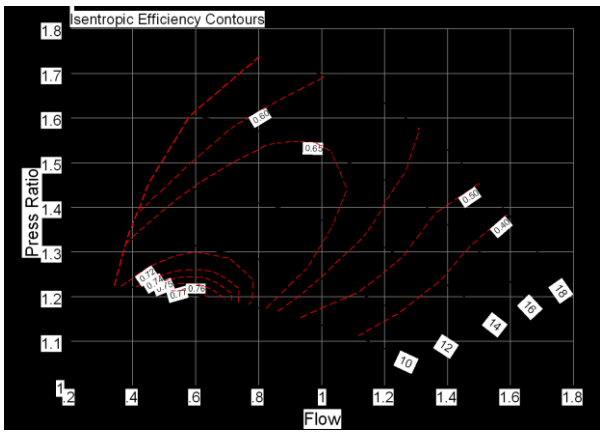


Figure 13: Compressor domain of use

Transformation into bong-graph components of other elements (Turbine, Control system, Cabin, Media …) have been studied by our partner CNRS Ampere and all modeling details and analysis can be found in [R07] and [R08] documents.

2.3 Additional components for Requirements modeling

Initially within the MODRIO library, the work of Dassault Aviation on requirements modeling has been mainly shared and tested through other libraries:

- 'DAProperties" library : to test new concepts and prototypes
- 'DA_Modelica_Requirements' and DLR 'Modelica_Requirements' libraries: to illustrate its needs in term of aircraft requirements formalization. Based on this work Aircraft requirements examples, and associated operators, were integrated within the 'Modelica_Requirements' library.

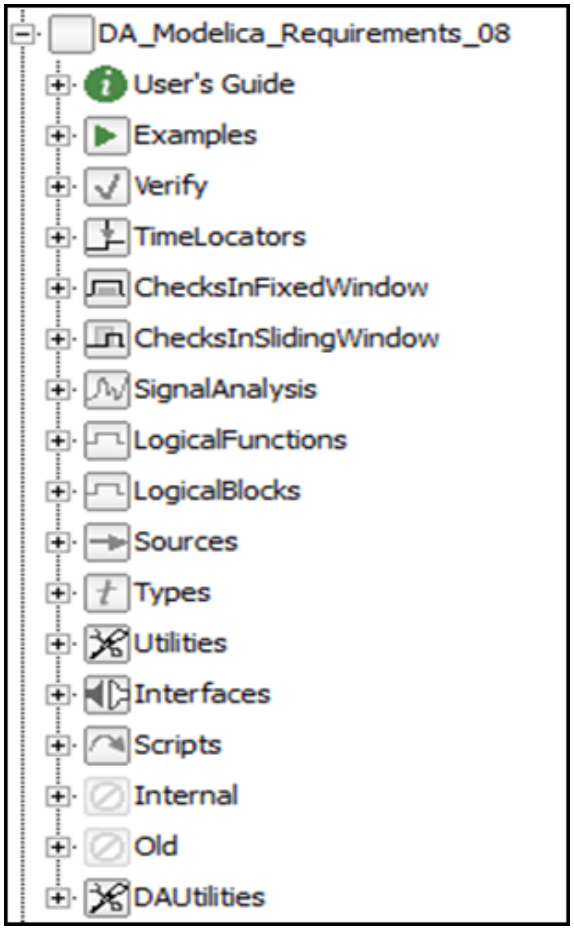


Figure 14 : Dassault Aviation Library Exchange

A detailed analysis of aircraft requirements was done to identify a typology of requirements. Then a synthesis [R06] was written using FORM-L and Modelica Standard Library, and elementary examples, as well as more complex aircraft examples were built.

2.3.1 Formal Operators

Dassault Aviation mainly worked on 'CheckInSlidingWindow', 'CheckInFixedWindow', 'SignalAnalysis', and 'LogicalBlocks' packages, defining operators to be able to:

1. Integrate along time
2. Integrate on a period of time
3. Check if the point X Y is inside a domain
4. Count events along time
5. Evaluate the duration of an event
6. Evaluate a frequency
7. Evaluate a derivative
8. Calculate an accumulation
9. Evaluate probabilities

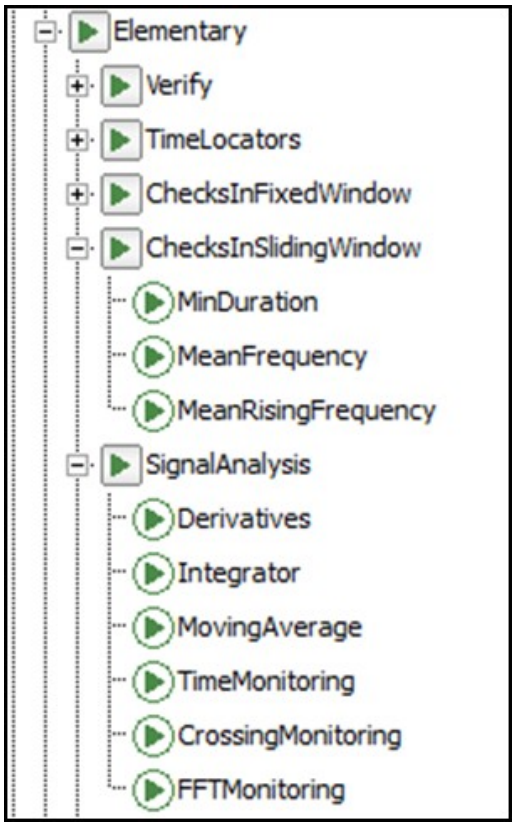


Figure 15 : Elementary Blocks Examples

Among 'SignalAnalysis' blocks, on prototype was built to calculate FFT (Fast Fourier Transformation). An FFT determines the frequency content and amplitudes of a sampled and periodic signal.

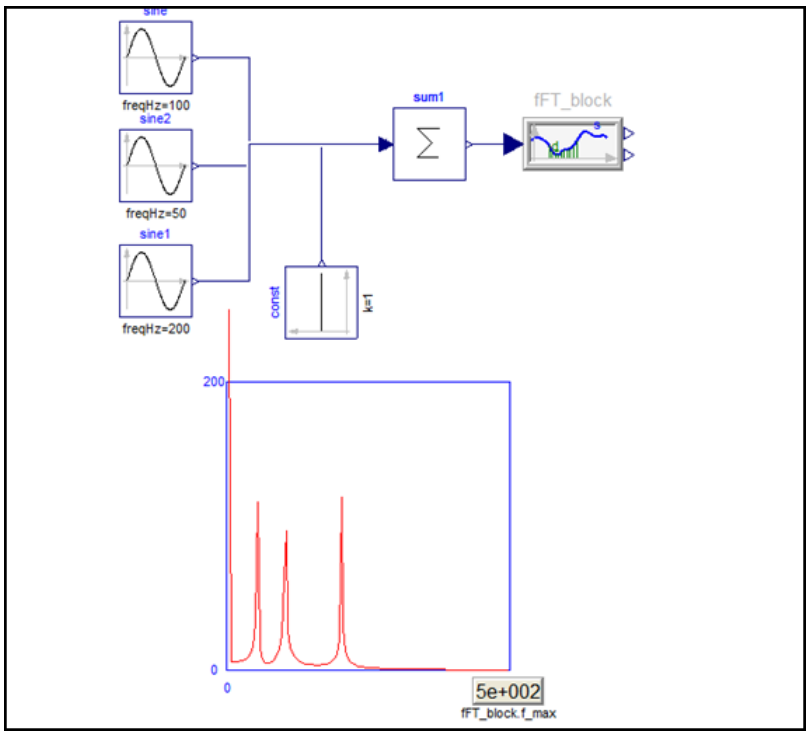


Figure 16 : FFT Monitoring Prototype

DLR brought improvements to this block and developed a new package within Modelica_Requirements library, containing all FFT blocks. It is now possible to calculate Eigen frequencies of an oscillating signal, such as the monitoring signal of a valve.

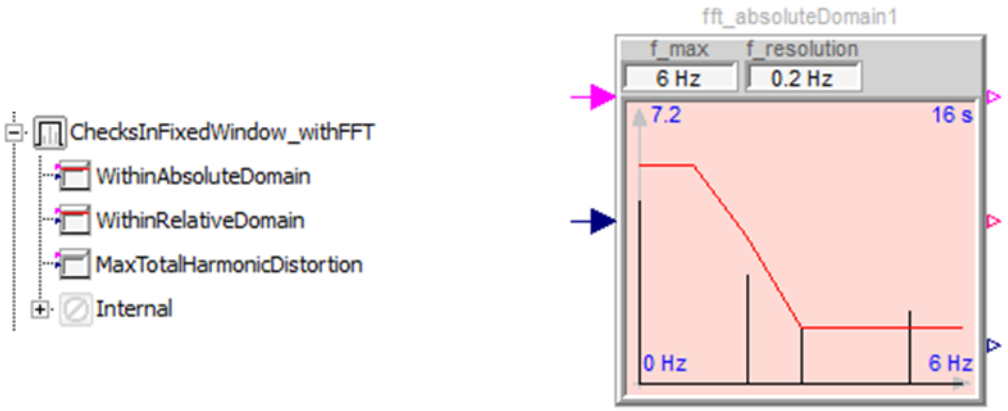


Figure 17 : FFT dynamically displayed in the icon of the block within Modelica_Requirements library

In 'SignalAnalysis' package, the following blocks were defined to calculate different type of derivatives. These operators are especially useful to express properties relating to rate of change (i.e inductor of an electrical circuit di/dt). In the 'CheckInSlidingWindow' package other operators were built to express property true in every sliding time window for instance.

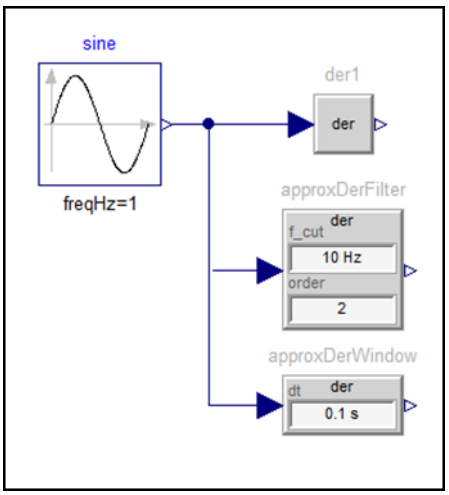


Figure 18 : Derivatives Blocks

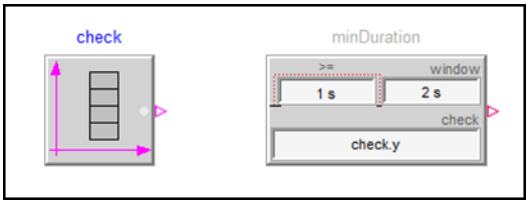


Figure 19 : MinDuration Block

2.3.2 Aircraft Requirements

After a detailed analysis of aircraft requirements, 12 types of requirements were identified and translated under Modelica:

1. *Threshold monitoring*
2. *Operating domain monitoring*
3. *Rate of change monitoring*
4. *Time integration monitoring*
5. *Oscillation monitoring*
6. *Monitoring with spatial or/and states condition*
7. *Monitoring with sets & array*
8. *Range monitoring*
9. *Monitoring with time condition*
10. *Monitoring with sequence of actions*
11. *Monitoring with probability condition*
12. *Assumptions*

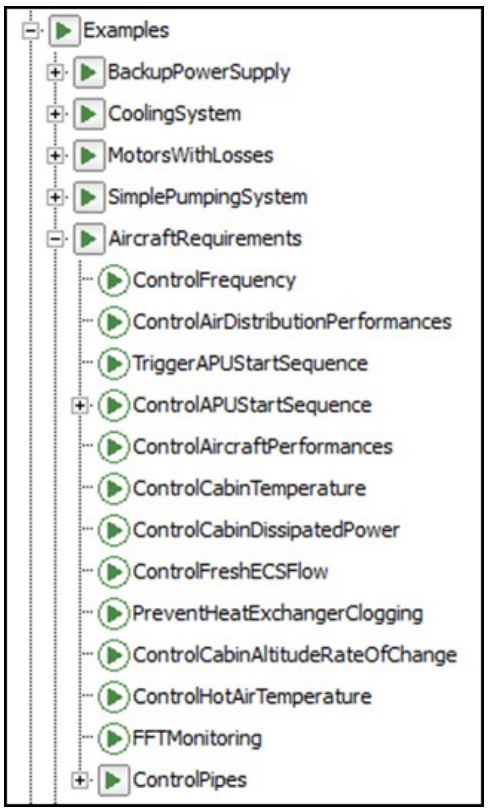


Figure 20 : Aircraft requirements examples

The following requirement is expressed to ensure passenger comfort. This property observed the period of time during which the cabin altitude rate of change is outside of a defined domain. The 'withinDomain', 'maxDuration' and 'der' blocks are used to check the validity of the requirement.

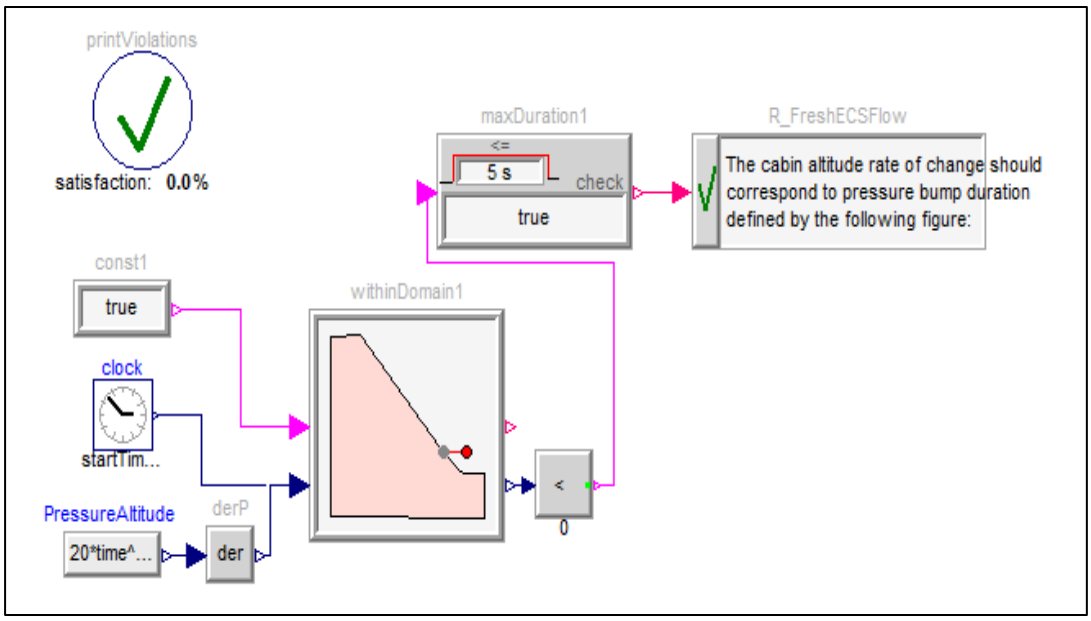


Figure 21 : Control Cabin Altitude Rate of Change Example

The following test illustrates the need to check the operations order of an Auxiliar Power Unit (APU) start sequence. This sequence is defined thanks to Modelica_Synchronous and observed thanks to 'after' block.

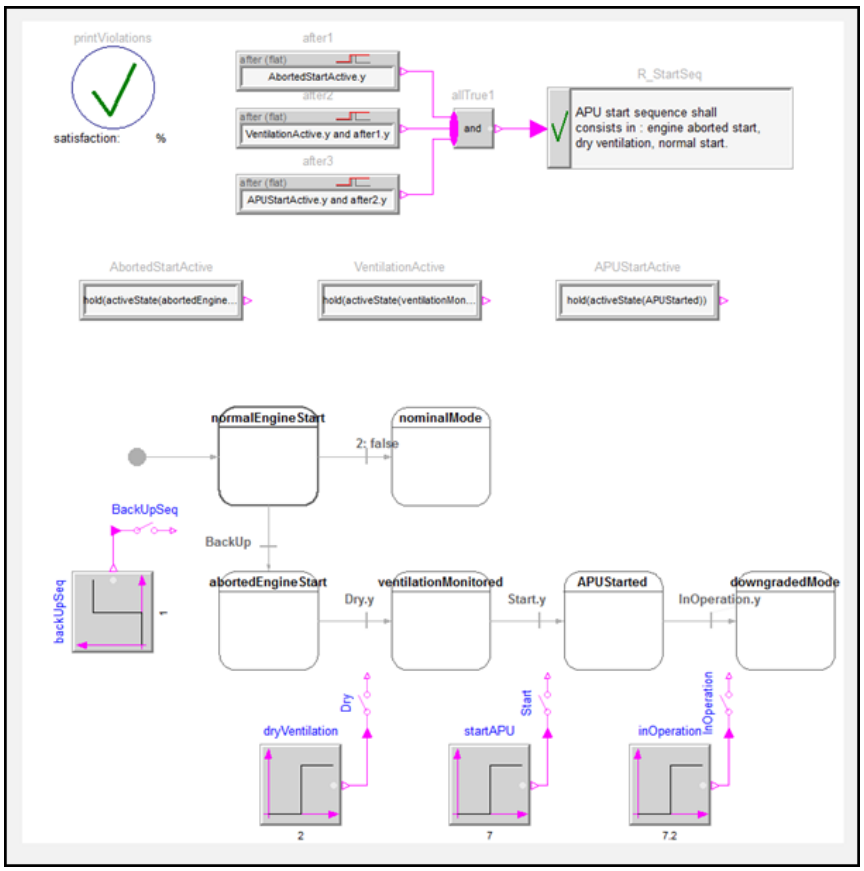


Figure 22 : Control APU Start Sequence Example

To prevent heat exchanger clogging it is important to calculate and check that the amount of ice is inferior to a maximal amount. To do that a 'TriggeredIntegrator' block was defined to integrate the product of air humidity and mass air flow along time.

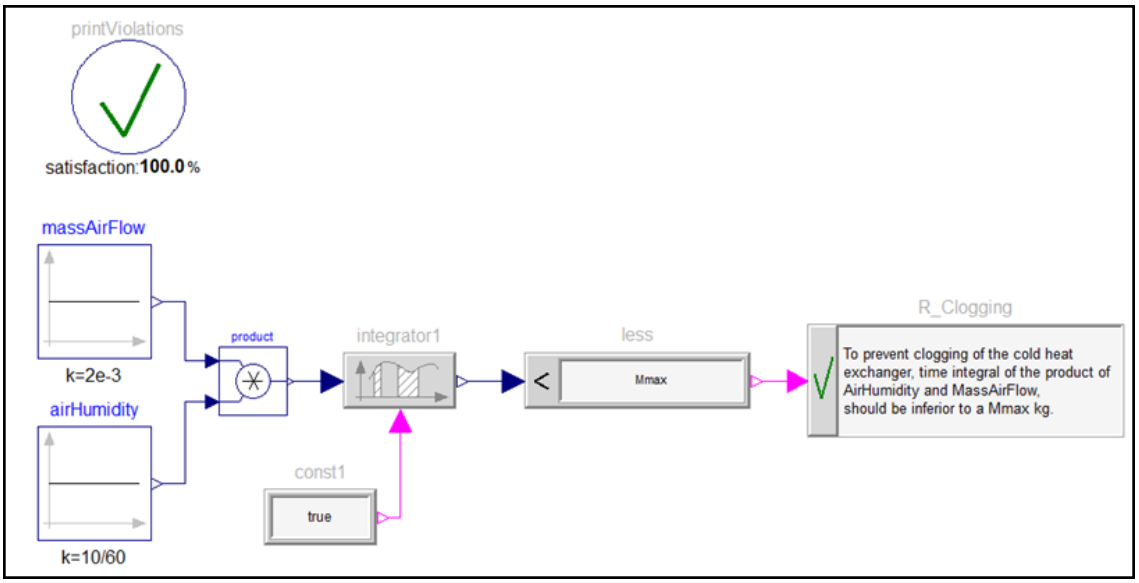


Figure 23 : Prevent Heat Exchanger Clogging Example

3. CONCLUSIONS

- Very important improvement have been done in MODRIO compared to the preliminary developments within Eurosyslib
- Types of requirements and associated operators were identified, shared and tested
- The Form-L specifications cover most of the needs identified in Aircraft Vehicle Systems requirements
- An important work have been done to graphically represent a requirement and easily identify requirements violation and global model satisfaction rate
- The open source library "Modelica_Requirements" developed by DLR now allows the user to model many of the requirements and assumptions he needs

4. REFERENCES

[R01] Modelica Conference 2015: "Towards Enhanced Process and Tools for Aircraft Systems Assessments during very Early Design Phase" Eric Thomas[1] Olivier Thomas[1] Raphael Bianconi[1] Matthieu Crespo[2] Julien Daumas[2]
[1]Dassault Aviation, France, [2]Liebherr Aerospace, France

[R02] Modelica Conference 2015: "Formal Requirements Modeling for Simulation-Based Verification" Martin Otter1, Nguyen Thuy[2], Daniel Bouskela[2], Lena Buffoni[3], Hilding Elmqvist4, Peter Fritzson[3], Alfredo Garro[5], Audrey Jardin[2], Hans Olsson[4], Maxime Payelleville[6], Wladimir Schamai[7], Eric Thomas[6], Andrea Tundis[5]
[1]Institute of System Dynamics and Control, DLR, Germany, Martin.Otter@dlr.de, [2]EDF, France, [3]PELAB, Linköping University, Sweden, [4]Dassault Systèmes AB, Sweden, [5]DIMES, University of Calabria, Italy, [6]Dassault Aviation, France, 7Airbus Group Innovations, Germany

[R03] WP2.1 / D2.1.1-Modelica extensions for properties modelling

[R04] WP2.1 / D2.1.2-Properties modelling method

[R05] WP2.1 / D2.2.1-Specification of Modelica extensions and interfaces for Bayesian networks, Fault trees and hybrid stochastic models

[R06] WP2.1 / DGT144425B Analysis of FORM-Language Applied to Aircraft Requirements, Maxime Payelleville - Dassault Aviation, France

[R07] WP6.1 / D6.1.1-DGT140458-Requirements for static model analysis

[R08] WP6.1 / Master Thesis « Modélisation et Analyse structurelle d'un ECS par approche Bond Graph »

[R09] MetaProperties_Modelica2015, Hilding Elmqvist[1], Hans Olsson[1], Martin Otter[2]
[1]Dassault Systemes AB Sweden, [2]Institute of System Dynamics and Control, DLR, Germany

[R10] DGT147579 - Properties2 Applied to Aircraft Requirements, Maxime Payelleville, Eric Thomas - Dassault Aviation, France

[R11] WP8.4 / D8.4.2- DGT153570-Business aircraft Demonstrator, Eric Thomas, Maxime Payelleville, Claire Campan, Emmanuel Ledinot – Dassault Aviation, France