Steinmann W.D., Zunft S.:

TechThermo - A Library for Modelica Applications in Technical Thermodynamics

2nd International Modelica Conference, Proceedings, pp. 217-224

TechThermo- A Library for Modelica Applications in Technical Thermodynamics

W.D. Steinmann, S. Zunft
German Aerospace Center (DLR)
Institute of Technical Thermodynamics
Pfaffenwaldring 38-40, 70569 Stuttgart
wolf.steinmann@dlr.de

Abstract

This paper describes the development of the Modelica library TechThermo. This library is intended as a basis for simulation projects in the area of technical thermodynamics. TechThermo enhances the efficiency of simulation activities by providing models describing essential processes which are not restricted to certain applications. The library contains connector-definitions, boundary conditions, basic descriptions of heat and mass transfer, control volumes, general property routines and components like turbines or heat-exchangers. The structure of models can be controlled by parameters thus providing the possibility to choose between different physical descriptions of a process.

Introduction

At DLR's Institute of Technical Thermodynamics various simulation activities cover projects reaching from solarthermal power generation to fuel cell systems. Initial experiences with Modelica show that the efficiency of the simulation activities can be improved significantly by definition of a base library containing models which are not restricted to a special application. By using such a base library developers working on differents projects share a common basis and can concentrate on the physical process which are typical for a special application while the common basis undergoes a continuous optimization resulting from the experience in the different projects.

Resulting from the former work with Modelica, some recommandations for improving the acceptance of a library can be made:

- models should be small and easy to understand
- models should be numerically robust
- the dependance on the right choice for the right initial conditions should be as small as possible
- the total number of components in a library should be limited; it's better to build several smaller libraries than building a single one containing everything
- the time period for developing a library should be limited
- an extensive application of object-oriented techniques like multiple inheritance doesn't improve readability

Experienced users should profit from TechThermo primarily by extending the models provided by the library thus minimizing the amount of trivial equations needed for describing a physical process. By standardization of connectors and variable names of input and output variables the reusability and readibility of models is enhanced. On the other hand, this library should allow a quick analysis of thermodynamic systems without the necessity of major modifications of the models. This also helps new users to apply Modelica for the simulation of thermodynamic systems.

1 Library Structure

TechThermo is organized in a four-level structure. There are seven main packages, which are stored in seperate files:

-	**Interface**	connectors and general base models
-	**Source**	boundary conditions
-	**Basic**	heat/mass transfer, control volumes
-	**Medium**	thermophysical properties
-	**Component**	basic components
-	**Subsystem**	simplified thermodynamic systems
-	**Example**	examples

Each of these main packages is divided in sub-packages. These sub-packages contain models which can be used without further modifications. If necessary, there are supporting models and data-records on the fourth level:

```
main package
        sub-package
                model 1
                model 2
                model 3
                package Support
                        support-model 1
                        support-model 2
                package Data
                        data record 1
                        data record 2
```

This structure enhances the orientation of the user within the library, since the first two levels contain only packages, while all models which can be used immediately are concentrated within the third level.

2 Control of Model Structure by Parameters

A general-purpose library has to offer a choice between different descriptions for a physical process to adapt the model to a certain simulation task. In Modelica flexibility can be reached by using replaceable models. In TechThermo, this approach is used to exchange models which show significant differences. In other cases, differences in a physical model only affect a single equation. Here, the usage of replaceable models seems not to be effective, since the total number of models will increase significantly. Instead, parameters are used in combination with if-expressions to influence the structure of a model. Two different cases can be distinguished: if there are only two alternatives, parameters of type Boolean are used to select the appropriate physical model. In TechThermo, names of such parameters start with *switch_* to distinguish them from other parameters:

```
model
        parameter Boolean switch_name = true;
....
equation
        if switch_name then
//      this part is only activated, if
//      switch_name== true
                equation 1;
                equation 2;
                equation 3;
                ...
        end if;
```

If there are more than two alternatives parameters of type Integer are applied. The names of these parameters start with *option_*. The alternatives are described in the info-section of a model. Users don't have to get along with names for different replaceable models, the total number of different models is kept small:

```
model
        parameter Integer opotion_name=1;
.....
 equation
        if option_name== 1 then
//      this part is only activated, if
//      option_name== 1
                equation 1;
                equation 2;
                equation 3;
                ....
        end if;
        if option_name== 2 then
//      this part is only activated, if
//      option_name== 2
                equation 1;
                equation 2;
                equation 3;
                ....
        end if;
        if option_name== 3 then
//      this part is only activated, if
//      option_name== 3
                equation 1;
                equation 2;
                equation 3;
                ....
        end if;
```

The switch_ and option_ parameters are declared with default values describing the standard case, so users can use the models without modifications of these parameters. Dymola translates only the parts which are activated by the parameters, other sections are ignored. As a result, control of model structure by parameters doesn't increase the calculation time. On the other side, modifications of switch_ and option_ parameters only become effective after a new compilation.

3 Connectors

The definition of connectors is an essential task during the creation of a base library. There's no unique ‚right' definition of connectors. In TechThermo four different kinds of connectors are defined:

mass-flow	m_dot [kg/s]	mass-flow rate
	h [J/kg]	spec. enthalpy
	p [Pa]	pressure
	x_i[-]	vector of mass-fractions
heat-flow	q_dot [W]	heat-flow rate
	t [°C]	temperature
thermal state	h [J/kg]	spec. enthalpy
	p [Pa]	pressure
	rho [kg/m³]	density
	s [J/kg/K]	spec. entropy
	t [°C]	temperature
	u [J/kg]	spec. internal energy
	x [-]	steam quality
	x_i [-]	vector with mass fractions
exergy-flow	exergy_dot [W]	exergy-flow rate

There are two connectors of each kind with different graphical presentation. The mass-flow connector allows the description of combined heat and mass-transfer. Specific enthalpy and pressure enable a description of the thermal state of the flowing medium. The description of a multi-component flow is possible by the vector x_i which contains the mass-fraction of each component. During the development of TechThermo different alternatives for the mass-flow connector have been discussed: the definition of individual mass-flow connectors for different transported media proves not to be efficient, since the number of connectors became to high. Instead, an universal mass-flow connector was defined. While the choice of mass-flow rate and pressure as connector variables is obvious, alternatives to specific enthalpy have been discussed. One disadvantage of specific enthalpy as a connector variable is that it is neither a 'through' nor an 'across' variable, a connection of more than two elements demands a mixing model. This might be avoided by using enthalpy-flow rate (product of specific enthalpy and mass-flow) as a connector variable. The major drawback of this concept is that information about the energy in the fluid is always related to the mass-flow rate: if the velocity of the fluid is zero, no information about the specific enthalpy of the fluid is available, if specific enthalpy is calculated from the enthalpy-flow rate division by zero must be avoided. Temperature as connector variable doesn't allow a clear definition of themal state in the two-phase region, entropy as connector variable is interesting from a theoretical point of view but is probably not accepted by all users.

The heat-flow connectors are used for thermal energy transfer without mass-flow, connector variables are temperature and heat flow rate.

The state connector transports information about the thermal state of a system. A characteristic of thermodynamics is that there are many different options for describing the themal state. Depending on the application, different sets of variables are preferred. While the steam quality is interesting when simulating a steam generator, it is unnecessary during the analysis of a gas turbine. A general library like TechThermo must offer a variety of state variables at the thermal state connector without urging the user to define all of them to avoid error messages. The state connector contains eight different kinds of state variables: spec. enthalpy h, pressure p, density rho, spec. entropy s, temperature t, spec. internal energy u, steam quality x and composition x_i. By using the model *NotUsedVariables* variables at the state connector can be equated with default values thus eliminating the necessity to define them explicitly. The choice of variables defined by *NotUsedVariables* depends on parameters of type Boolean.

The set of connectors is completed by the general exergy-connectors which transport exergy-flows like mechanical power or electricity without further specification of the kind of power-flow.

3.1 Adapters to Connectors of Modelica Standard Library

The Connector main package contains a sub-package with models to join models from the Modelica Standard Library to models from TechThermo. These adapters include a connector from each of these two libraries and define the relation between the variables of the two connectors.

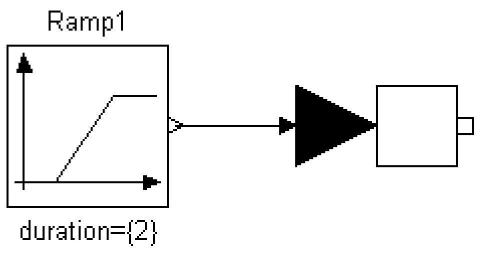

Fig. 1: Adapter-model *OutHeatFlowInSignal* transforming signal to thermodynamic connector variable

Fig. 1 shows the adapter-model *OutHeatFlowInSignal* that connects a heat flow connector to the connector *InPort* from the Modelica Standard Library. The adapter is used to transform a general signal provided by model *Ramp* to a thermodynamic variable. The thermodynamic variable is either temperature or heat-flow rate depending on the value of parameter *option_defsignal* in *OutHeatFlowInSignal.* Similar adapters are implemented for the other connectors in TechThermo.

3.2 TwoPort Models

Many processes in technical thermodynamics can be regarded as systems with an inflow and an outflow. As base classes for models TechThermo provides for all defined connectors models with an inflow and an outflow connector. There are two different kinds of models: while *TwoPortCM* only includes two connectors with different graphical representation, *TwoPort* extends *TwoPortCM* introducing also simple relations for the corresponding variables of the two connectors. These relations can be controlled by parameters. Many processes affect only some of the connector variables while others remain constant. By using *TwoPort* as a base model, the necessity to define explicitly connector variables which remain constant is avoided by setting the corresponding parameter values. For example models describing heat transport can extend the *TwoPort*-model for heat flow:

```
partial model TwoPortCM
" partial model heat flow element with two
connectors"
//------------connectors--------------------
       In InHeatFlow;
       Out OutHeatFlow;
end TwoPortCM ;
```

```
model TwoPort
"model heat flow element with two connectors"
 extends Support.TwoPortCM;
//------------boolean switches for additional
//            equations---------
parameter Boolean switch_q_dot_const=false
"if switch_q_dot_const=true then
q_in_dot+q_out_dot=0";
parameter Boolean switch_t_const=false
"if switch_t_const=true then t_in=t_out";

//Internal variables

flow SIunits.HeatFlowRate q_in_dot;
SIunits.CelsiusTemperature t_in;
flow SIunits.HeatFlowRate q_out_dot;
SIunits.CelsiusTemperature t_out;

equation

       q_in_dot = InHeatFlow.q_dot;

       t_in = InHeatFlow.t;

       q_out_dot = OutHeatFlow.q_dot;

       t_out =  OutHeatFlow.t;

//relations between connector variables at
//heat_cut1 and heat_cut2 dependant on boolean
//parameters:

       if switch_q_dot_const then

         0.0 = q_in_dot + q_out_dot;

         //heat flow rate remains constant

       end if;

        if switch_q_dot_const then

         t_in = t_out;

         //temperature remains constant

         end if;

     end TwoPort;
```

The introduction of variables like t_in, q_in_dot, t_out, q_out_dot shorts the names of connector-variables.

There are also *TwoPort* mass-flow elements with additional connectors for heat flow or exergy flow. These models offer the option to activate a stationary energy balance.

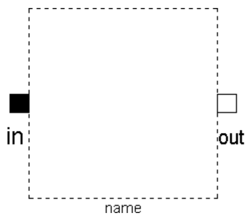

Fig. 2: Icon for *TwoPort*-model for heat-flow

4. Source Models

The Source package contains models representing boundary conditions in thermodynamic systems. Boundary conditions can be introduced for all kind of connectors. The connector variables used as boundary conditions are selected by parameters thus avoiding the necessity to define seperate models for various kinds of boundary conditions.

The following Modelica-Code shows the definition of a source for heat flow:

```
model ParameterDefined
heat-flow source with optional definition of
heat-flow variables by parameters"

TTInterface.HeatFlow.Out OutHeatFlow;


//------------parameters----------------------
//switches for variables at connectors:
parameter Boolean witch_q_dot_def=false
if switch_q_dot_def=true, OutHeatFlow.q_dot is
determined by parameter q_dot_para";
parameter Boolean switch_t_def=false
"if switch_t_def=true, OutHeatFlow.t is
determined by parameter t_para";

//values for variables at outlet if
//corresponding switch-parameter=true
parameter SIunits.HeatFlowRate q_dot_para=1.0
"value for heat-flow rate HeatFlowOut.q_dot at
outlet if switch_q_dot_def=true";
parameter SIunits.CelsiusTemperature
t_para=25.0
"value for temperature at HeatFlowOut.t at
outlet if switch_t_def=true";
equation
        if switch_q_dot_def then
          OutHeatFlow.q_dot = q_dot_para;
         end if;

         if switch_t_def then
          OutHeatFlow.t = t_para;
         end if;

end ParameterDefined;
```

This model allows to select a constant temperature or a constant heat flow (or both) as boundary conditions. In addition to models providing constant boundary conditions, there are also models with an signal input offering the possibility to control one of the boundary variables by an external signal source.

5. Models for Heat and Mass Transfer

Package Basic offers basic models for describing heat and mass transfer processes. There are models describing thermal conduction, convective heat transfer and radiation heat transfer. Depending on switch-parameters values for heat conductivity, heat transfer coefficient or emissivity can be assumed as being constant or not. The model *HeatTransfer* extends the *TwoPort*-model for heat flow and calculates convective heat-transfer assuming a constant value for the heat transfer coefficient. There's no storage of energy in the element, so parameter *switch_q_dot_const = true*, i.e. *q_in_dot = -q_out_dot.* The temperature difference between the inlet and the outlet is calculated from the heat-flow rate, the area and the heat transfer coefficient:

```
model HeatTransfer
 "basic model describing heat transfer "
extends TTInterface.HeatFlow.TwoPort
(switch_q_dot_const=true);

SIunits.CoefficientOfHeatTransfer alpha_trans;
parameter SIunits.CoefficientOfHeatTransfer
alpha_trans_const=1000
"if switch_alpha_const==true then
alpha_trans=alpha_trans_const";
parameter SIunits.Area surface_area=1.0
"area normal to direction of heat transfer";

//-------switch-parameter------------------
parameter Boolean switch_alpha_const=true
"if switch_alpha_const==true then
alpha_trans=alpha_trans_const";

equation

if switch_alpha_const then
        alpha_trans = alpha_trans_const;
end if;

q_in_dot = surface_area*(t_in -
t_out)*alpha_trans;

end HeatTransfer;
```

The model *HeatTransferVariable* extends the more basic *HeatTransfer*-model, parameter switch_alpha_const being set to false and the heat transfer coefficient alpha_trans is now calculated from the properties of the fluid and the velocity. There are two connectors for the mass-flow determining the heat transfer coefficient. An additional state connector is needed to introduce the density of the fluid for calculating the velocity from mass-flow rate. *HeatTransferVariable* offers various possibilities for calculating the heat transfer coefficient, the user can choose the apporopriate physical model by setting an option_parameter.

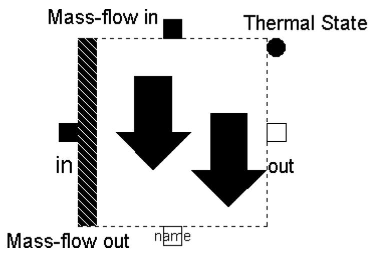

Fig. 3 model *HeatTransferVariable* with connectors for mass-flow and thermal state to calculate heat transfer coefficient.

Similar to the models describing heat transport, the models included in TechThermo for calculating pressure loss due to mass flow offer the possibility to choose between different physical model depending on flow geometry, accuracy and velocity range.

6 Control Volumes

A proper description of control volumes is essential for modelling the dynamic behaviour of a thermodynamic system. In TechThermo, the dynamics of a finite mass with a single connector is described by the model *ThermalCapacity*. The temperature of the system is calculated from the heat flowing into or out of the system. Heat capacity and mass of the system can be determined by parameters *c_heat*, *m_const,* but there's also the possibility to extend the model and introduce other definitions for mass or heat-capacity after a modification of the corresponding switch_parameters. This might be useful if other parameters are preferrred (example: pipe with mass calculated from diameter and density) .

```
model ThermalCapacity
"control volume finite thermal capacity"

       TTInterface.HeatFlow.In HeatCut
       SIunits.Mass m;
       // mass of thermal capacity
       SIunits.SpecificHeatCapacity c_heat;
       // specific heat capacity

       parameter SIunits.Mass m_const=1
       "const. value mass if
       switch_m_const==true";
       parameter SIunits.SpecificHeatCapacity
       c_heat_const=500
        "const. value heat-capacity if
       switch_c_heat_const==true";

       //      Boolean switches
       parameter Boolean switch_m_const=true
       "if switch_m_const==true then
       m=m_const";

       parameter Boolean
       switch_c_heat_const=true
      "if switch_c_heat_const==true then
      c_heat=c_heat_const";


    equation

      if switch_c_heat_const then
       //      specific heat capacity is
       //     defined by parameter
        c_heat = c_heat_const;
      end if;

      if switch_m_const then
        //      mass is defined by parameter
        m = m_const;
      end if;

      //      transient energy conservation
      HeatCut.q_dot = m*c_heat*der(HeatCut.t);


    end ThermalCapacity;
```

A general description for a finite volume with a single mass-flow connector and a heat-flow input is provided by the model *ControlVolume* including energy and mass conservation:

```
model ControlVolume
      parameter Integer n_comp=1;
      parameter SIunits.Volume v_control=1;


      //-connector for thermal state---
      TTInterface.ThermalState.In
      StateCut(n_comp=n_comp);

      //connector for inflow or outflow of mass
      TTInterface.MassFlow.In
      InMassFlow(n_comp=n_comp);


      //---------- connector for heat
      //transferred to ControlVolume----
        TTInterface.HeatFlow.In InHeatFlow;



    protected
        SIunits.InternalEnergy energy;
        SIunits.MassFlowRate m_inflow_dot;
        SIunits.MassFlowRate m_outflow_dot;
    equation

      InMassFlow.p = StateCut.p;
      InMassFlow.x_i = StateCut.x_i;
      InHeatFlow.t = StateCut.t;

        energy =
        v_control*StateCut.u*StateCut.rho;

        der(energy) = m_inflow_dot*InMassFlow.h
        + InHeatFlow.q_dot +
        m_outflow_dot*StateCut.h;

      v_control*der(StateCut.rho) =
      InMassFlow.m_dot;

      m_inflow_dot = if InMassFlow.m_dot <= 0
      then 0.0 else InMassFlow.m_dot;
      m_outflow_dot = if InMassFlow.m_dot > 0
      then 0.0 else InMassFlow.m_dot;

    end ControlVolume;
```

Model *ControlVolume* gets information about the density of the fluid by the state connector *StateCut*, usually by an external property routine providing a relation between internal energy u, density rho and pressure p, so the model is not restricted to a certain kind of fluid. The mass-flow can either be positive or negative, the direction also influences the specific enthalpy at the mass-flow connector.

7 Thermophysical Properties

In technical thermodynamics the calculation of physical properties plays an important role. Since TechThermo should be a general purpose library, the routines included for property calculation are mainly based on univeral physical models, descriptions optimized for a certain medium are not applied. For most models, the characterization of a substance by molar mass and critical point properties is sufficient. With respect to calculation time and errors usually introduced by simplified physical models in other parts, the reduced accuracy of general property models seems to be acceptable.

In TechThermo the property models are seperated from other model parts, informations are exchanged by a state connector. Connecting a property model usually presents the final step in creating a model. The general models for property calculation are organized in four sub-packages according to the physical condition:

- Solid
- Liquid
- Gas
- MultiPhase

Descriptions for the solid and liquid state offer the possibility to take variations of density into account or not. Depending on the demanded accuracy, one of the following models describing the gas state can be selected:

- *PerfectGas*:
  pv=RT, specific heat capacity is constant
  low pressure, small temperature variations, far from saturation temperature
  example: dry air at ambient conditions

- *IdealGas*:
  pv=RT, specific heat capacity is temperature dependant
  medium pressure, significant temperature variation, far from saturation temperature
  example: air in gas turbine

- *RealGas:*
  Redlich Kwong equation:
  $$p = \frac{RT}{v-b} - \frac{a}{T^{0,5} v(v+b)}$$
  coefficients a and b are calculated from critical values,
  heat capacity is temperature dependant
  example: superheated water-steam

The Redlich-Kwong equation is a cubic equation of state. In order to determine the correct solution, a cubic equation solver based on the method of Cardano is applied. The calculation of caloric values like spec. enthalpy or entropy demands an equation for the specific heat capacity dependant on temperature. Since no general models are available, polynomes (usually second degree) must be provided for calculating the specific heat capacity.

The model *SaturationTemperature* calculates the saturation temperature from the pressure according to Antoine, the model *EvaporationPressure* calculates the evaporation enthalpy. Together with the models describing liquid and gaseous state the calculation of thermophysical properties with a minimum of medium specific data is possible.

8 Components

Main package *Component* contains models for the fundamental units of a system in technical thermodynamics like compressors, turbines, heat-exchanger, pipes, valves, tanks or burners. These models are composed of general basic models and a specific property routine. For example the general model for a turbine is model *Basic* (located in package Turbine.Support):

```
model Basic
"turbine without specification of working
fluid"
        extends TTInterface.MassFlow.TwoPort(
          final switch_m_dot_const=true,
          final switch_h_const=false,
          final switch_p_const=false,
          final switch_x_i_const=true);

       //----exergy-connector mechanical power
       provided by expansion
      TTInterface.ExergyFlow.Out PmechCut;

       //---------connector for
       //thermodynamic properties at inlet
        TTInterface.ThermalState.In InletState;

       //---------connector for thermodynamic
       //properties after ideal expansion-----
        TTInterface.ThermalState.Out
       IdealExpansionState;

        //---------parameters
       parameter SIunits.Efficiency
       eta_expansion_const=0.8
      "const. efficiency of turbine if
      switch_eta_expansion_const==true";

      //---------switch-parameters----
      parameter Boolean
      switch_eta_expansion_const=true
      "if switch_eta_expansion_const==true
then eta_expansion=eta_expansion_const";


     protected
       SIunits.SpecificEnthalpy dh_ideal;
       // spec. enthalpy difference isentropic
       //expansion
        SIunits.SpecificEnthalpy h_out_ideal;
       // spec. enthalpy after isentropic
       //expansion
        SIunits.Efficiency eta_expansion;
```

```
    equation

      if switch_eta_expansion_const then
        eta_expansion = eta_expansion_const;
      end if;

      InletState.h = h_in;
      InletState.p = p_in;
      InletState.x_i = x_in_i;

      //      thermal state after isentropic
      //expansion from p_in to p_out
       //      is defined by p2 and entropy
      //entropy_inlet_cut.s
      IdealExpansionState.p = p_out;
      IdealExpansionState.s = InletState.s;
      IdealExpansionState.x_i = x_in_i;
      h_out_ideal = IdealExpansionState.h;

//      decrease specific enthalpy isentropic
//expansion
        dh_ideal = h_out_ideal - h_in;
//      specific enthalpy after real expansion
       h_out = h_in + dh_ideal*eta_expansion;
//      mechanical power provided by expansion
        PmechCut.exergy_dot = m_in_dot*(h_out -
        h_in);

    end Basic;
```

In this basic version of turbine calculation, the outlet condition is calculated from the inlet condition by assuming an isentropic expansion and multiplying the difference in spec. enthalpy by the efficiency of the turbine. The calculation demands the knowledge of the entropy at the inlet, which is provided by state connector *InletState* and the thermodynamic state after the ideal expansion which is provided by connector *IdealExpansionState.* In this basic version, the efficiency of the turbine is determined by the parameter *eta_expansion_const*; a more elaborate turbine model might be implemented by extending *Basic,* setting *switch_eta_const* = false and introducing a calculation procedure for the efficiency *eta_expansion*.

The model should now be used for expanding air, the model *AirTurbine* extends first the model *TurbineMC,* which contains two mass-flow connectors, an exergy-flow connector for the mechanical work delivered by the turbine and the icon presentation of the turbine. The general tubine model *Basic* is connected to the outer connectors, finally the model is completed by joining two property routines for air to the thermal state connectors *IdealExpansionState* and *InletState* of *Basic.*

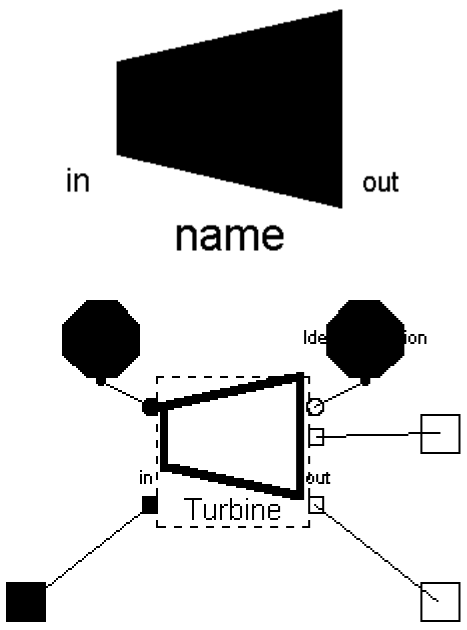

Fig.4 : Icon for model *AirTurbine* and internal view with basic turbine model and two property models.

9 Subsystem

The package subsystem contains simplified models for thermodynamic systems like solar collectors or cyclic processes. These models should be used to complete systems when the main interest of the simulation doesn't focus on the part modelled by the subsystem-component.

10 Current Status and Conclusion

The development of TechThermo started during a project dealing with fuel-cell systems and has taken advantages of the experiences gained in this area. After an initial period of theoretical considerations almost all components identified as essential are implemented. TechThermo is now applied in two different projects, one dealing with high temperature thermal storage, the other dealing with the dynamics of solarthermal steam generation. Compared to first experiences with Modelica for simulating thermodynamic systems, the efficiency of the modelling activities could be improved significantly thus showing the importance of a common base library. At the moment TechThermo undergoes a continuous improvement due to the practical experiences in the projects. While the total number of models in TechThermo should remain more or less constant, future activities will concentrate on the improvement of the numerical stability of the components.