# ON MODELING OF HEAT EXCHANGERS IN MODELICA

**Sven Erik Mattsson**

Department of Automatic Control
Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
E-mail: SvenErik@control.LTH.se

## ABSTRACT

It is demonstrated how Modelica™ is used in an application to develop models that are useful when solving real problems. Modelica is a new unified modeling language being developed in an international effort to promote object-oriented and non-causal modeling, and exchange of model libraries. The application is a heat exchanger where the media are liquids, typically water. This type of heat exchangers can be used for district heating of houses and for production of hot tap water. The model developed illustrates very nicely the power of Modelica. The modularization concepts support flexible model components which are easy to use and to adapt when making a model of a real system with heat exchangers. The concept of class parameters support medium parameterization and arrays of model components support discretization. The expressive power of Modelica allows complete listings of the developed model components to be given. The model produces simulation results that agree very well with measured data.

## INTRODUCTION

A new language called Modelica[1] for physical modeling is developed in an international effort. The main objective is to develop a new unified modeling language which promotes exchange of models and model libraries and which supports non-causal modeling with true ordinary differential and algebraic equations and the use of object-oriented constructs to facilitate reuse of modeling knowledge.

This paper demonstrates how Modelica is used in a specific application to develop models that are useful when solving real problems. The task is to develop a model for a heat exchanger where the media are liquids. The model developed is good enough to be used to solve real problems but simple enough to allow a detailed explanation in a conference paper. The paper discusses major issues such as decomposing the physical description of the heat exchanger from the properties of the media in a powerful way. The concepts of Modelica will be described and explained when needed in the modeling process. For more information on Modelica see Elmqvist and Mattsson (1997b), Elmqvist and Mattsson (1997a) and `http://www.Dynasim.se/Modelica/`.

[1]Modelica is a trade mark of the Modelica Design Group

The application is a heat exchanger CB50 which is a part of the TS30 unit from Alfa-Laval Thermal for heating and production of hot tap water for a one-family house. Figure 1 contains a schematic diagram where the radiator part is left out. In the primary circuit flows hot district heating water, which is used to heat the water in the secondary circuit to produce hot tap water.

A heat exchanger model which describes the relations between the pressures, the rates and the temperatures of the flows at the four ports of the heat exchanger is to be developed. For clarity the focus is on heat exchangers where the media are single medium liquids. Basic relations which can be found in good standard textbooks, e.g., Holman (1972) are used to describe behavior. Validations against measured data from a TS30 heat exchanger unit show a very good agreement between measured and simulated behavior.

Heat exchangers are very common components in chemical process systems and energy systems including power generation and heating and ventilation. To minimize energy losses, it is important to have optimal systems, where both static and dynamic properties of the components are adjusted to each other. Typically the static behavior of a heat exchanger is well-known, while the dynamical behavior of a system of connected valves, pumps and heat exchangers is less known. Thus, there is a need for mathematical models and dynamic simulation.

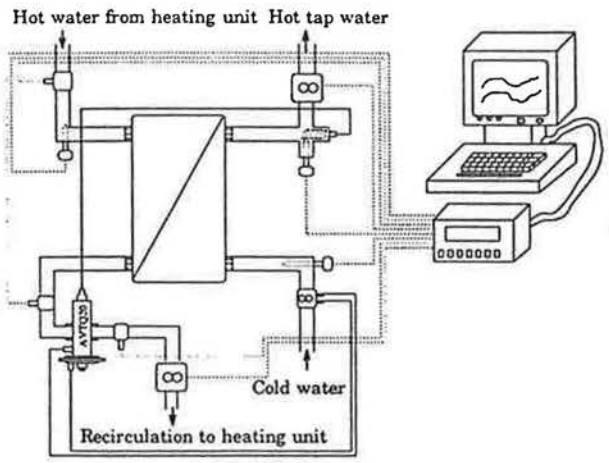


**Figure 1** A schematics of the heat exchanger unit TS30 with a measurement logging system.

## BASIC MODEL STRUCTURE

The first step of the object-oriented approach when modeling a component is to identify the ways in which it can interact with its environment. The behavior of a component can then either be described in terms of equations or as interconnected components. The modeling approach is hierarchical.

A heat exchanger has two ducts with two ports each. These four ports constitute interfaces to which other components can be connected as shown in Figure 1.

### Variable types

The essence of physical modeling is to describe behavior in terms of mathematical relations between physical quantities. In Modelica the type for pressure quantities is declared as

```
type Pressure = Real(Unit = "Pa");
```

where `Real` is the name of a predefined type. A real variable has a set of attributes such as unit of measure, minimum value, maximum value and initial value.

To simplify the use of Modelica and to support compatibility, there is an extensive standard library of type definitions which is always available with every Modelica translator. The type definitions in this base library are based on ISO 1000 and its naming conventions for physical quantities.

### Connectors

Interaction between model components is described by connecting *connectors* in Modelica and can graphically be represented by lines. A Modelica model defined graphically will contain graphical information specifying positions of submodels and connectors, paths for connections and graphical icon representations. Modelica defines a format for graphical annotations to make icons and model diagrams portable. All the graphical information is left out in the listings presented in this paper. Interactive browsing tools to list textual representations should be able to filter out graphical information to make the listings shorter and more transparent.

Defining a set of connector classes is a good start when a library for a new application domain of is to be developed. A common set of such definitions, used in all components in the library, promotes compatibility.

The status at a port of the heat exchanger can be described by one pressure, one temperature and one volume flow rate, if single medium liquid flows are assumed and if there is no interest in describing pressure, temperature or velocity profiles over the port area. To support this, a connector class is defined as

```
connector FlowInlet
  Pressure           p;
  flow VolumeFlowRate q;
  Temperature        T;
end FlowInlet;
```

A connection between two connectors means that the components having the same names are connected to each other down to the levels of simple quantities. The natural meaning of connecting quantities such as pressure, temperature, voltage and position, is that they should be equal, while for flow rates, currents, forces, torques etc. the physical meaning is that they should sum to zero. A sum-to-zero equation is generated when the prefix `flow` is used in the connector definition. In Modelica the convention is that a flow into the component is counted as positive.

### The interface

The Modelica model

```
partial model HEXShell "Base class for
      heat exchanger and section models"
  FlowInlet A1, A2, B1, B2 "Ports";
  parameter HEXParameters Pars;
  virtual model LiquidA = BasicLiquid;
  virtual model LiquidB = BasicLiquid;
end HEXShell;
```

outlines the interface part of a heat exchanger model. The keyword `partial` indicates that this model class is incomplete. At various places such as between the name of a class and its body or after a component declaration it is allowed to have a string. It is treated as a comment attribute and is meant to be a documentation that tools may display in special ways. Due to space limitations comments will be left out in the following.

The model declares four ports: `A1`, `A2`, `B1` and `B2`. The model will not assume any specific port to be inlet or outlet. Which ports that are inlets and which ports that are outlets depend on the actual flows. The two ducts of the heat exchanger will be called `DuctA` and `DuctB`. See Figure 2. The two ports of `DuctA` will be referred to as `A1` and `A2` and the two ports of duct `B` will analogously be referred to as `B1` and `B2`. It will be assumed that the ports `A1` and `B1` are at one end of the heat exchanger and that the ports `A2` and `B2` are at the other end. For example, if there is a flow from `A1` to `A2` and a flow from `B1` to `B2` then the heat exchanger is operated as a parallel flow heat exchanger. But if the flow in `DuctB` is from `B2` to `B1` then the heat exchanger is operated as a counter flow heat exchanger.

`HEXShell` declares a structured parameter `Pars` to describe physical properties such as dimensions. The

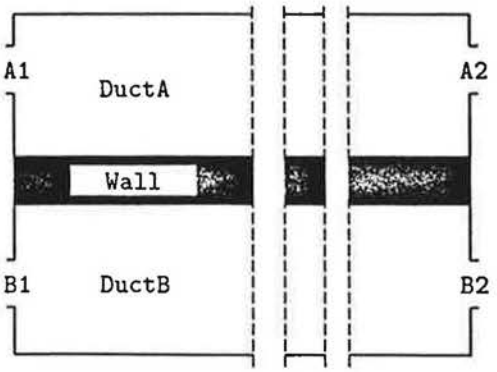


**Figure 2** A conceptual picture of a heat exchanger.

keyword parameter specifies that Pars is constant during a simulation experiment, but can change value between experiments. The idea is to make it simple for a user to modify the behavior of a model. The physical parameters of the heat exchanger will be identified along the derivation of the behavior.

## LIQUID MODELS

The heat exchanger model needs models of the liquids flowing in the two ducts. The model HEXShell defines two *model classes parameters*, LiquidA and LiquidB, for describing the properties of the media in the two ducts. The idea is to have a well defined interface between the description of the heat exchanger and the media to make it easy to modify the heat exchanger to cope with different media and descriptions of different complexity. The medium models are to be used in the duct models. On the heat exchanger level there is no use of medium models, but to make it easier for the user to change medium models, the classes of the medium models are made parameters of the heat exchanger model.

The default class BasicLiquid specified by the model can be declared as

```
model BasicLiquid
  Pressure p;
  Temperature T;
  Density rho;
  SpecificHeatCapacity c;
end BasicLiquid;
```

Each actual liquid model must at least contain the attributes p, T, rho and c, since an actual model must contain all public components of the default class. Note that an actual model need not be constructed by inheritance from the default class. Such a requirement would have made it difficult to use models developed at various places. There is just a requirement on agreement of the names of the components.

In the future there will hopefully be standard models available in Modelica for many media in the same way as there are data bases for physical properties today. Medium models may actually be implemented as interfaces to such databases.

For water it is often sufficient to use constant values for density and heat capacity. Such a model can be declared in the following way:

```
model BasicWaterModel =
  BasicLiquid(rho = 1000, c = 4180);
```

It means that rho and c become constant with the given values. It is not possible to change these values interactively between simulation runs. To allow this the model must be declared as

```
model BasicWaterModel1 = BasicLiquid(
  redeclare parameter Density rho = 1000,
  redeclare parameter SpecificHeatCapacity c=4180
);
```

which turns rho and c into parameters and sets default values, which may be changed interactively between simulation runs.

It is important to include information that makes it possible to check the validity of a model. A simple approach is to specify ranges for the variables. The model for water is not valid when the water is freezing to ice or boiling to steam. For normal pressures, the condition can be expressed in terms of the temperature, which should be between 273 K and 373 K. To include this condition in our model we modify it as

```
model BasicWaterModel2 =
  BasicLiquid(T(Min = 273, Max = 373),
              rho = 1000, c = 4180);
```

A more general condition can be included in the equation section by specifying an assertion as

```
assert condition;
```

It is simple to make a liquid model in which the density is temperature dependent. It is just to include the relation and possibly declare some parameters. A model which assumes a linear expansion of volume with temperature may be declared as

```
model BasicLinearLiquid
  extends BasicLiquid;
  parameter Temperature T0;
  parameter Density rho0 "at temperature T0";
  parameter CubicExpansionCoefficient alphaV;
equation
  rho = rho0/(1 + alphaV*(T-T0));
end BasicLinearLiquid;
```

The statement "extends BasicLiquid" means that the model BasicLinearLiquid inherits all properties of BasicLiquid. The model BasicLinearLiquid specifies in addition three parameters and a affine relation between density and temperature.

A model for water is specified as

```
model BasicLinearWaterModel =
  BasicLinearLiquid(T(Min = 273, Max = 373),
                    T0 = 293, rho0 = 998,
                    alphaV = 0.00018, c = 4180);
```

## BEHAVIOR DESCRIPTION

Heat exchangers may internally look very different, but the basic idea is to let two media, which flow on the two sides of the wall, exchange heat through the wall without being mixed. A heat exchanger consists from a conceptual point of view of two ducts with a common wall through which heat can flow. See Figure 2. The model will be based on this conceptual view.

The complexity of using partial differential equations to describe the heat transfer from the hot side to the cold side can be avoided, since we do not have the ambition to describe the internal behavior of the heat exchanger for example in order to calculate thermal stresses. A common approach which divides the heat exchanger into a number of slices or sections along the direction of the flow will be taken. Such a section consists of two duct sections and one wall section.

## A DUCT SECTION MODEL

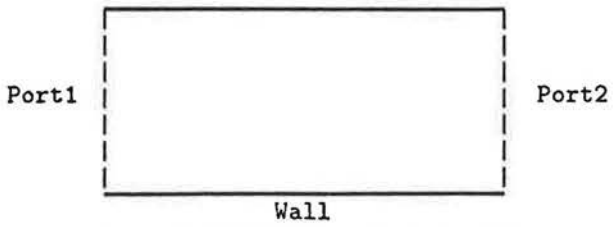


**Figure 3** The connectors of a duct section.

A Modelica model for a duct section is given in Listing 1. As indicated in Figure 3 three connectors are needed. These are defined first in the model `HEXDuctSection`. There are `Port1` and `Port2` for the liquid flow and `Wall` to describe interaction with the wall. Then the model defines a parameter record, `Pars`, and a class parameter, `Liquid`, for the liquid properties in order to make it easy to exchange medium models.

Let us consider the behavior description and discuss the equations in turn.

Assume that the duct is filled all the time and that the liquid is incompressible. Then the mass balance degenerates to the fact that the volume inflow rates at the two ports must sum to zero.

$$q_1 + q_2 = 0 \tag{1}$$

The model `HEXDuctSection` uses `Port1.q` to represent the volume inflow rate, $q_1$, and `Port2.q` to represent $q_2$. Thus Equation (1) becomes `Port1.q + Port2.q` in the model `HEXDuctSection`.

The pressure drop over the duct is modeled as

$$p_1 - p_2 = \frac{\rho}{C_v^2}|q_1|q_1 \tag{2}$$

where $p_1$ and $p_2$ are the pressures at the two ports, $\rho$ is the density of the liquid and $C_v$ is a constant.

The heat balance of a duct gives

$$\frac{dH}{dt} = \Phi_w + \Phi_1 + \Phi_2 \tag{3}$$

where $H$ is the enthalpy. The left hand side of the balance equation represents the change of the thermal energy in the duct. The first term of the right hand side, $\Phi_w$, is the rate of the heat flow across the wall into the duct and $\Phi_1$ and $\Phi_2$ are the changes of thermal energy of the duct due to the flow through the two ends of the duct;

$$\Phi_i = c_i\rho_i q_i T_i, \quad i = 1,2 \tag{4}$$

where the factors of the right hand side represent properties of the flow at the port; $c_i$ is the specific heat capacity, $\rho_i$ is the density, $q_i$ is the volume flow rate and $T_i$ is the temperature.

The enthalpy is given by

$$H = c\rho VT \tag{5}$$

```
connector WallConnector
 VolumeFlowRate q;
 Temperature T1, T2, T;
 flow HeatFlowRate Phi;
end WallConnector;

record DuctParameters
  Volume V;
  Area  Cv;
end DuctParameters;

model HEXDuctSection
 FlowInlet Port1, Port2;
 WallConnector Wall;
 parameter DuctParameters Pars;
 virtual model Liquid = BasicLiquid;
 Liquid L1 "The liquid at Port1.";
 Liquid L2 "The liquid at Port2.";
 Temperature T;
 Density rho;
 SpecificHeatCapacity c;
 Enthalpy H;
equation
 // Hydraulics
 Port1.q + Port2.q = 0;
 Port1.p - Port2.p =
   rho/Pars.Cv^2*abs(Port1.q)*Port1.q;
 // Thermodynamics
 der(H) = Wall.Phi +
   L1.c*L1.rho*Port1.q*Port1.T +
   L2.c*L2.rho*Port2.q*Port2.T;
 H = rho*Pars.V*c*T;
 T =   if Port1.q > 0 then Port2.T else Port1.T;
 rho = if Port1.q > 0 then L2.rho  else L1.rho;
 c =   if Port1.q > 0 then L2.c    else L1.c;
 // Communication
 Wall.q  = Port1.q;  Wall.T  = T;
 Wall.T1 = Port1.T;  Wall.T2 = Port2.T;
 L1.p = Port1.p;     L2.p = Port2.p;
 L1.T = Port1.T;     L2.T = Port2.T;
end HEXDuctSection;
```

**Listing 1** A model of a duct section.

where $V$ is the volume of the duct and $T$ is a representative mean temperature of the liquid. The specific heat capacity, $c$, and the density, $\rho$ are to be taken at that temperature. The temperature at the outlet will be assumed to be $T$;

$$T = \text{if} \quad q_1 > 0 \quad \text{then} \quad T_2 \quad \text{else} \quad T_1 \tag{6}$$

Validations against measured data indicate that this is a valid approximation so it seems not useful to introduce more dynamics or delays from $T$ to the temperature at the outlet. The approach to view $T$ to be the mean value of the temperatures at the port, $T = (T_1 + T_2)/2$, has the drawback that an increase of the temperature at the inlet implies an instantaneous decrease of the temperature at the outlet, since $T$ due to the dynamics does not respond instantaneously.

The last four lines set up the interaction with the wall and the liquid models.

## A HEAT TRANSFER MODEL

A Modelica model for a wall section is given in Listing 2. First, it defines two connectors WA and WB for the interaction with the ducts. The meaning of the parameters and the other quantities will be explained below when we discuss the equations in turn.

The model of the heat transfer through the wall focuses on the heat flow across the wall. Heat flow along the wall is neglected as well as the heat capacity of the wall, which means

$$\Phi_A + \Phi_B = 0; \tag{7}$$

The heat flow rate across the wall is modeled as

$$\Phi_B = \Delta T_m / R \tag{8}$$

where $R$ is the overall thermal resistance between the two ducts and $\Delta T_m$ is a suitable mean temperature difference between the two ducts along the wall.

To obtain a model that has a statically correct behavior, the common approach is to take $\Delta T_m$ as the *log-mean temperature difference*, $\Delta T_{lm}$, defined as

$$\Delta T_{lm} = \frac{\Delta T_1 - \Delta T_2}{\ln(\Delta T_1 / \Delta T_2)} \tag{9a}$$

where $\Delta T_1 = T_{A,1} - T_{B,1}$ and $\Delta T_2 = T_{A,2} - T_{B,2}$ are the temperature differences at the two ends of the duct. The formula is badly conditioned if $\Delta T_1 \approx \Delta T_2$ so when $|\Delta T_1 - \Delta T_2| < 0.05 \max(|\Delta T_1|, |\Delta T_2|)$ it is better to use

$$\Delta T_{lm} = 0.5(\Delta T_1 + \Delta T_2) \times \left(1 - \frac{1}{12}\frac{(\Delta T_1 - \Delta T_2)^2}{\Delta T_1 \Delta T_2}\left[1 - \frac{1}{2}\frac{(\Delta T_1 - \Delta T_2)^2}{\Delta T_1 \Delta T_2}\right]\right) \tag{9b}$$

which gives a relative error which is less than $10^{-5}$. Equation (9) is implemented a bit more elaborately to avoid divisions by zero. A good model compiler will detect that the expression $(\Delta T_1 - \Delta T_2)^2/(\Delta T_1 \Delta T_2)$ appears twice and eliminate multiple evaluations at simulation.

The thermal resistance, $R$, between the hot and cold side can be decomposed into four terms

$$R = R_A + R_w + R_B + R_f \tag{10}$$

where $R_A$ and $R_B$ are the thermal contact resistances between the liquid in the ducts and the wall, $R_w$ is the thermal resistance of the wall, $R_f$ is the thermal fouling resistance due to deposits and dirt on the wall.

The contact resistances between a liquid and the wall are modeled as

$$\begin{aligned} R_i &= (h_i A_w)^{-1}, \quad i = A, B \\ h_i &= h_0 \left|\frac{q_i}{q_0}\right|^{n_h} [1 + a_h(T_i - T_0)], \quad i = A, B \end{aligned} \tag{11}$$

where $h_i$ is the surface coefficient of heat transfer, $A_w$ is the area of the common wall between the ducts, and

```
record ThermalSurfaceParameters
 SurfaceCoefficientOfHeatTransfer h0;
 VolumeFlowRate q0;
 Real n;
 LinearThermalCoefficient ah;
 Temperature T0 "nominal mean";
end ThermalSurfaceParameters;

record WallParameters
 Area A;
 Thickness d;
 ThermalConductivity lambda;
 ThermalSurfaceParameters SA, SB;
 Real Y "corrugation factor";
 ThermalResistance Rf "Fouling";
end WallParameters;

model HEXWallSection
  WallConnector WA, WB;
  parameter WallParameters Pars;
protected
  ThermalResistance R, RA, RB, Rw;
  Temperature DTlm, DT1, DT2;
equation
  // Heat transfer
  WA.Phi + WB.Phi = 0;
  WB.Phi = DTlm/R;
  // Log-mean temperature difference
  DT1 = WA.T1 - WB.T1;   DT2 = WA.T2 - WB.T2;
  DTlm =
    if (abs(DT1-DT2) > 0.05*max(abs(DT1),abs(DT2)))
    then (DT1-DT2)/ln(DT1/DT2)
    else if DT1*DT2 == 0 then 0.5*(DT1+DT2)
    else 0.5*(DT1+DT2)*
         (1-sqr(DT1-DT2)/(DT1*DT2)*
          (1 + sqr(DT1-DT2)/(DT1*DT2)/2)/12);
  // Thermal resistance
  R = RA + Rw + RB + Pars.Rf;
  RA = 1/(Pars.SA.h0*abs(WA.q/Pars.SA.q0)^Pars.SA.n*
          (1+Pars.SA.ah*(WA.T-Pars.SA.T0))*Pars.A);
  RB = 1/(Pars.SB.h0*abs(WB.q/Pars.SB.q0)^Pars.SB.n*
          (1+Pars.SB.ah*(WB.T-Pars.SB.T0))*Pars.A);
  Rw = Pars.d/(Pars.lambda*Pars.Y*Pars.A);
end HEXWallSection;
```

**Listing 2** A model of the heat exchanger wall.

$h_0$, $n_h$, $q_h$ and $a_h$ are constants to be identified from measured data and $T_0$ is an estimation of the mean value of lowest and highest appearing temperatures.

The thermal resistance of the wall, $R_w$, is calculated as

$$R_w = \frac{d}{\lambda Y A_w} \tag{12}$$

where $d$ is the thickness of the wall, $\lambda$ is the thermal conductivity of the wall material and $Y$ is a correction factor for the corrugation of the wall.

The model HEXWallSection has been implemented as a primitive model, where the behavior is described directly in terms of equations. When more elaborate models are to be developed it is recommendable to decompose the model into thermal resistors in series.

```
record HEXParameters
  DuctParameters DuctA, DuctB;
  WallParameters Wall;
end HEXParameters;

model HEXn
  extends HEXShell;
  parameter Integer n "Number of sections";
  HEXDuctSection
    DuctA[n](Pars(V  = Pars.DuctA.V/n,
                  Cv = sqrt(n)*Pars.DuctA.Cv),
             redeclare model Liquid = LiquidA),
    DuctB[n](Pars(V  = Pars.DuctB.V/n,
                  Cv = sqrt(n)*Pars.DuctB.Cv),
             redeclare model Liquid = LiquidB);
  HEXWallSection
    Wall[n](Pars = WallPartPars(fraction = 1.0/n,
                                Wall = Pars.Wall));
equation
  connect(A1, DuctA[1].Port1);
  connect(B1, DuctB[1].Port1);
  for i in 1:n-1 loop
    connect(DuctA[i].Port2, DuctA[i+1].Port1);
    connect(DuctB[i].Port2, DuctB[i+1].Port1);
  end for;
  connect(DuctA[n].Port2, A2);
  connect(DuctB[n].Port2, B2);
  for i in 1:n loop
    connect(DuctA[i].Wall, Wall[i].WA);
    connect(DuctB[i].Wall, Wall[i].WB);
  end for;
end HEXn;
```

**Listing 3** The base model of a heat exchanger.

## A HEAT EXCHANGER MODEL

A model of a heat exchanger can now be defined by putting together a number of duct section models and wall section models. See Listing 3.

The record type `HEXParameters`, which is referred by the shell model `HEXShell`, is defined as the aggregation of the parameters of the two ducts and the wall.

The model `HEXn` uses `HEXShell` as a base class and specifies in addition two arrays of duct section models and one array of wall section models, which are of the same length, parameterized by `n`. The equation section connects the pieces to each other and to the ports of `HEXn`.

The declarations of the component models include hierarchical parameter propagation to set default values for the parameters of the elements. Some of the parameters have to be recalculated, since the section parts have other physical dimensions than the heat exchanger itself. To explain how this is done, let us in more detail discuss one of the declarations, for example

```
HEXDuctSection
  DuctA[n] (Pars(V  = Pars.DuctA.V/n,
                 Cv = sqrt(n)*Pars.DuctA.Cv),
            redeclare model Liquid = LiquidA);
```

The declaration sets the two simple parameters `Pars.V` and `Pars.Cv` individually in terms of the parameters of `HEXn`. The construct `V = Pars.DuctA.V/n` means that the default value of `DuctA[i].Pars.V` should be set to `Pars.DuctA.V/n` for $1 \leq i \leq n$. Note, that the right hand side is resolved in the scope of `HEXn`. The declaration also sets an actual model `LiquidA` to the virtual model `Liquid`.

The wall has more parameters, where all but two parameters should just be copied when propagated to its sections. It is convenient to use a function for this since it allows multiple assignments. A function declaration is similar to a class declaration, but it starts with the keyword `function`. The input arguments are marked with the keyword `input` and the result arguments of the function are marked with the keyword `output`. Functions have an algorithm section instead of an equation section. The algorithm section should contain ordered assignment statements, if-then-else constructs and loops.

```
function WallPartPars
  input  Real fraction;
  input  WallParameters Wall;
  output WallParameters WallPart;
algorithm
  WallPart := Wall;
  WallPart.A := fraction*Wall.A;
  WallPart.Rf := Wall.Rf/fraction;
end WallPartPars;
```

All parameters are copied in the first statement. Then two of them are modified.

## VALIDATION OF THE MODEL

When this is written in August 1997, there is yet no Modelica translator which can transform the heat exchanger model developed above to a representation which can be simulated. There is a translator that can handle a subset of Modelica, but it cannot handle arrays of components.

The mathematical model described above has earlier been implemented in the object-oriented language Omola [Mattsson *et al.* (1993), Andersson (1994)] and simulated in the interactive environment OmSim. The Omola model was validated against measurements from a heat exchanger of type CB50 from Alfa-Laval Thermal. A description of the experiments and comparisons done can be found in Ericsson and Östberg (1993). The result of the validation is that there is a good agreement between simulated and measured behavior. The Omola model and the result of the validation have also been published in Mattsson *et al.* (1994).

To indicate that the mathematical model developed above is good, we will here show a typical result from that validation. In the experiment the flow of hot water from the district heating unit was kept constant with a rate of 0.25 l/s and the consumption of tap water was varied as shown in Figure 4. For CB50 the volume of each duct is 0.094 liters and the area of the heat transporting wall is 1.1 $m^2$. The measured time series of the tap water flow was used as input to a simula-

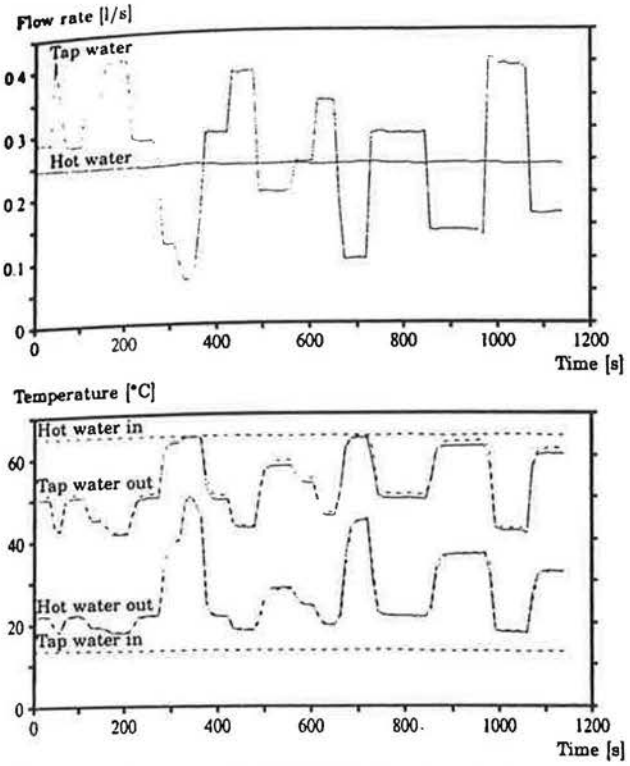


**Figure 4** Result of a validation experiment. The upper figure shows measured flow rates and the lower figure shows measured temperatures (dashed lines) and simulated temperatures (solid lines).

tion model where the heat exchanger was modeled by three sections. The simulated and measured temperatures at the outlets are shown in Figure 4. The agreement between measured and simulated temperatures at the outlets are very good. The static error on the hot side may depend on bad positioning of the temperature sensor. All temperature sensors were calibrated carefully before the experiments.

## CONCLUSIONS

An object-oriented model of a heat exchanger unit operating under normal conditions has been developed. The model has been validated against measurements from a real system and the agreement between measured and simulated behavior is very good.

Use of the new general object-oriented modeling language Modelica has been illustrated and discussed in detail for an important technical system. It has been demonstrated how Modelica promotes flexible model components. In particular it is demonstrated that Modelica indeed supports decomposition of media properties from physical properties of the heat exchanger. It is very easy to change media or the complexity of media models.

It is not feasible to make a model that describes all situations equally well. In normal operation of a heat exchanger, the flows do not change directions. The model developed can from a structural point of view handle flows that change directions. However, all behaviour descriptions are not good for small flows. For example, the model for the heat transfer between a liquid and the wall describes the behaviour for a forced flow at stationary conditions. It means that the model does not predict any heat transfer for zero flow rate. However, the use of object-oriented ideas makes it possible to easily modify the model when the equations are available.

Modelica is designed to support reuse of model knowledge. The ideal world is that a user who wants to solve a real problem finds a ready-made model in his model library. The second best option is that he is able to develop the desired model by just putting together components from the model library by using a graphical editor. This paper focuses on development of such library components for heat exchangers. The need for flexible and reliable model components means that a developer of such a model library must define general and parameterized components and he or she must also provide redundant information such as declaration of variables to allow automatic consistency checking. Modelica supports this task very well.


## Acknowledgements

The work has been supported by the Swedish National Board for Technical Development under project P9304688 "Modeling and simulation of complex systems" and by Sydkraft under project 391 "Modeling and control of energy processes". The provision of a heat exchanger system for experiments from Alfa Laval Thermal is gratefully acknowledged.


## REFERENCES


ANDERSSON, M. (1994): *Object-Oriented Modeling and Simulation of Hybrid Systems.* PhD thesis ISRN LUTFD2/TFRT--1043--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H. and S. E. MATTSSON (1997a): "An introduction to the physical modeling language Modelica." In *Proceedings of the 1997 European Simulation Symposium (ESS'97).* The Society for Computer Simulation, Passau, Germany.

ELMQVIST, H. and S. E. MATTSSON (1997b): "Modelica — The next generation modeling language, An international design effort." In *Proceedings of the 1st World Congress on System Simulation.* Singapore.

ERICSSON, M. and P. ÖSTBERG (1993): "Dynamisk provning av värmeväxlarsystem (Dynamic testing of heat exchanger systems)." Master Thesis ISRN LUTFD2/TFRT--5490--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

HOLMAN, J. P. (1972): *Heat Transfer*, third edition. McGraw-Hill Book Company.

MATTSSON, S. E., M. ANDERSSON, and K. J. ÅSTRÖM (1993): "Object-oriented modeling and simulation." In LINKENS, Ed., *CAD for Control Systems*, pp. 31–69. Marcel Dekker, Inc., New York.

MATTSSON, S. E., M. ERICSON, and P. ÖSTBERG (1994): "An object-oriented model of a heat-exchanger unit." In *Proceedings of the European Simulation Multiconference, ESM'94*, pp. 297–303. SCS, The Society for Computer Simulation, Barcelona, Spain.