AN NMF-BASED COMPONENT LIBRARY FOR FIRE SIMULATION

Kjell Kolsaker
University of Trondheim
The Norwegian Institute of Technology
Division of Heating and Ventilation
N-7034 Trondheim

ABSTRACT

The simulation of the transient behaviour of buildings is becoming more important as faster and cheaper computers reach the market. Many simulation programs and specialized tools have been developed to simulate complex situations. With growing problem complexity the simulation programs have to provide more user support and more advanced models. Ibis is usually solved by writing tailored application programs using specialized solution methods and menus or window interfaces. The results are efficient tools for the end user.

The development of specialized end-user applications is normally very time-consuming. Therefore good programs are normally only available for the most common-used calculations. Simulation models that differ from the standard models in any matter have to be either made by running an existing model using "tricks", or using a general modular simulation program combining existing or self-written modules into the desired system.

Common modular simulation programs for building simulation include for example TRNSYS, HVACSIM + and IDA, these all providing flexibility in writing own models, in principle. However, the use of this kind of program require a lot of knowledge and experience of the simulation program.

An example of a problem that is not completely covered by specialized programs is simulation of smoke distribution in a ventilation system with mechanical ventilation during the development of a fire. A few initial simulations have been carried out using IDA with a limited library of specialized components. Developing models for this kind of programs is often a quite laborious task.

This paper presents a practical application of a definition language called NMF, Neutral Model Format. (Sahlin 1989) applied to the simulation of fire development in buildings using the IDA modular simulation program developed by ITM (Institute of Applied Mathematics) of **Stockholm (BRING** 1990). The project has been run for a short time during the winter 1990-91 at SINTEF Applied Thermodynamics in Trondheim, Norway, and includes model development, testing and running. The model library will be the kernel of a component library to be used by other simulations later, and will be further developed as the need for new facilities occur.

This work shows that NMF is already working well as a concept for model development for practical applications. The subsequent exploration of the possibilities of this definition format is interesting, and further projects should be initiated.

1 THE PROBLEM

Smoke movement has proved to be the major killer when fire in a building. From the fireroom smoke is spread uncontrolled via gaps, open doors etc. as well as via ductwork as a part of the mechanical ventilation system, to other parts of the building. Research activity the latest year has proved that smoke movement via ductwork like this can be eliminated by keeping the mechanical ventilation running even after fire is detected. This philosophy implies that fire dampers should not be used as before.

Updated calculations of smoke movement in a building when fire via ductwork when ventilation system is switched off have shown that fire dampers with fusible link will not close in the early stage of a fire, due to too low temperature.

None of the existing programs for fire simulations examined were found fully suited for the modelling of this spesific problem. Examples are RVENT, developed at SINTEF Applied Thermodynamics/ NBL (Norwegian Fire Research Laboratory), FIREX (NBL) and FAST, NBS National Bureau of Standards (NIST). As a matter of this the necessary models in IDA were worked out. After some initial model development, calculations like this are made possible. Results have been correlated to fullscale measurements.

The problem includes the un-linear behavior of the pressure drops in a ductwork with mechanical ventilation combined with air expansion caused by the increasing temperature.

A second problem is to model the transient heat flow in building constructions and ventilation ducts. Heat storage in the surface layers of the walls influence much on the room temperature in the initial phase of a fire.

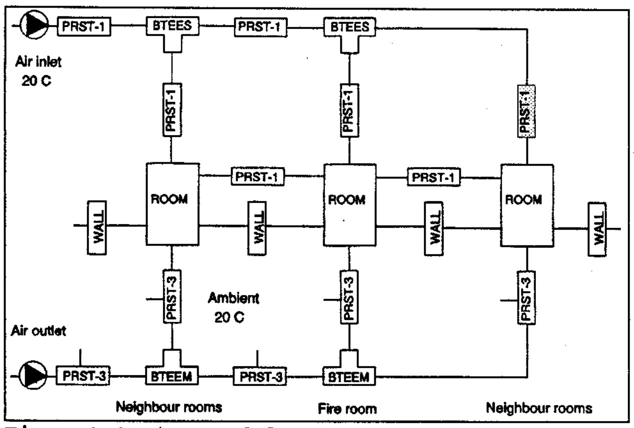
A third problem is to model the heat transfer in the fire room. The problem is a combination of convective and radiative heat transfer and could only be modelled properly with for example a three dimensional finite difference (FD) model solving the governing thermodynamic and fluid-dynamic equations. Such simulations have been applied to the same problem with the FD-program KAMELEON (Holen 1989).

The room model used by IDA is a single zone model. The systen is modelled with constant radiative and convective heat transfer coefficients. Wall area and or resulting film coefficient have been correlated to measurements to compensate for air stratification and horizontal temperature gradients in the room.

A problem included in the model library is air contaminant concentrations in all parts of the system. These simulations have not been carried out yet.

2 THE MODEL, A REVIEW

We have modelled a part of a typical hotel building under fire conditions in one of the rooms. In the initial simulations we are interested in the temperature in the outlet duct system. In the next simulations (which we have not yet done) we are interested in the smoke gas concentrations in the neighbouring rooms and the outlet ductwork.



Figur 1 System model.

The model consists of three rooms (Figur 1). The room in the middle is the fire room. Each room is connected to a common air inlet and air outlet duct. Between each room there is an air leakage, modelled by pure pressure resistances.

The walls are single components including one heat transfer resistance with combined convection and radiation at each side. This very rough approach, excluding many important effects in a room under fire conditions, is used as a first approach.

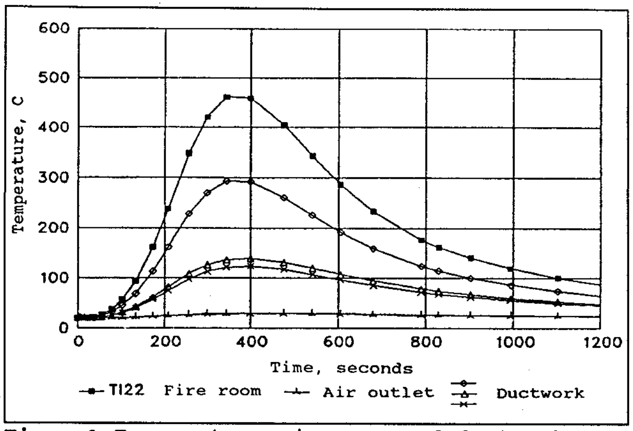
The fire itself is not modelled yet, just given as a table of measured heat flow. By including stoichiometric reactions and the dynamics in fire development, an simple model for the fire can be included later.

Air flow is uni-directional. This is the simplest approach,and care must be taken so that air flows the right way during the whole simulation. Bi-directional air flow will introduce improved flexibility to the models, and it opens the possibility to simulate real conditions where air flow turns the other way under a fire.

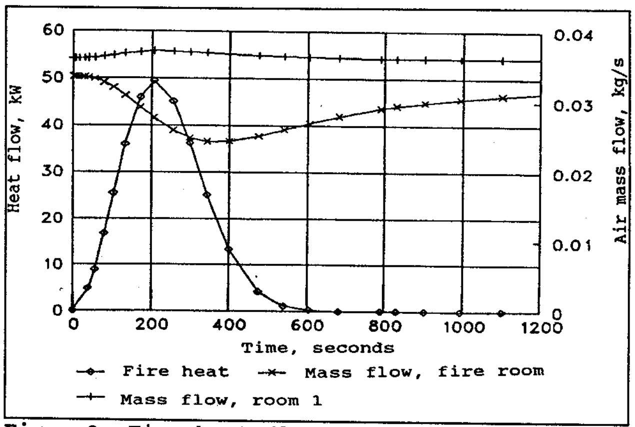
3 SIMULATION RESULTS

The objectives of the simulations is to analyze the effect thermal conditions in the outlet ductwork. An example of simulated temperature development in the ductwork is presented in Figur 2.

The simulated temperatures in the room (Figur 2) reaches 470 C. The temperature at the outlet has only a small temperature increase because of duct wall heat loss and cold air mixing from the other rooms.



Figur 2 Temperatures in room and ductwork.



Figur 3. Fire heat flow and air mass flow.

Some major heat flow and mass flow values are presented in Figur 3. The fire heat load reaches 400 kW at time 250 seconds. Mass flow in the fire room decreases nearly 50 %. Mass flow in the neigbour room has a small increase due to the decreased air flow from the fire room.

4 A BRIEF REVIEW OF THE NMF

NMF is a definition format for continuous and discontinuous models, not bound to one special simulation program. The format includes equation description, assignments, and description of the variables and parameters used in the model. Arrays are provided for flexibility and can be applied both to boundaries and internal model discretization. Subroutines can be defined in regular programming language and be used as a part of the equations or initial parameter processing. Definitions can be either local or global, letting several models use the same definitions. The following key principles (Sahlin 1989) are followed, ensuring generality:

1. Continuous models are expressed in terms of equations
2. Variables and interconnections are typed
3. Large models allow hierarchical decomposition
4. Validation is integrated into the modelling process

The Neutral Model Format has been developed at ITM in Stockholm with collaboration from LBL, and is currently being tested and evaluated. The evaluation runs parallel with defining and testing new models for IDA. Model authors are using NMF as the main modelling tool, and the production of Fortran subroutines is a one way process from NMF to Fortran. Waiting for an automatic Fortran translator, the manual translation process is made straight forward by standardized naming and implementation rules. It has rarely been needed to let the Fortran code deviate from the NMF model description; occasional demands for changes or extensions have led to fruitful discussions.

The result is a definition format that works well today, and a range of components and component families is already defined. Some of the model families that are implemented or are under development are presented beneath:

- Basic models for heat transfer and air movements in buildings (ITM)
- Basic models for district heating sub-central including pipes, valves, heat exchangers and controllers (SINTEF) (Kolsaker 1990).
- The models from the IEA ANNEX 17 (ITM)
- Models for buildings and ventilation systems (Ingegerd Ljungkrona, Chalmers)
- Basic models for electric networks and control components for test use (ITM)

We will here present a documentation of the parts of NMF that are used in the presented component library. For details not included here please refer to (Sahlin 1989) and (Bring 1990). A complete syntax for NMF is also worked but not published yet.

4.1 Basic Parts of a Model

The central part of an NMF model is a set of equations under a section heading EQUATIONS. The equations have the format

<expression> = <expression>

where the expressions are written in a natural manner, well known from higher programming languages Fortran or Pascal. Conditional expressions are allowed, as well as standard floating point functions defined in Fortran F77.

The equations define relations between variables, time derivatives of variables, parameters, and constants.

The entities used in the equations are declared and typed, partly in the model, partly in a global section, as described briefly below.

Links define how a model may be connected to neighboring models or to an environment of a system of models. They usually correspond to physical ports of a physical component, e.g terminals of a pipe, surfaces of a wall, etc. All variables that connect a model to its surrounding must appear on these links, there are no 'global variables'. Links are described further below.

4.2 Global Definitions

Variables, parameters, and links are typed in a global declaration section. These global declarations are then referenced from the model definitions. This feature is essential to support development of reusable component models that are compatible with each other. Models can be combined into libraries, sharing common link types; individual models can freely be replaced by variants, as long they use the same link types.

Constants can be defined in the global declaration section. The global definition file for the presented component library is described beneath:

```
VARIABLE_TYPES
/* name          unit           kind */
   massflow      "kg/s"         THRU
   temp          "C"            CROSS
   rel_pres      "kPa"          CROSS
   fraction      "kg/kg"        CROSS
   heatflux      "W"            THRU
   mass          "KG"           CROSS

LINK_TYPES
/* name          variable types... */
   t_q           (temp,heatflux)
   p_m_t_x       (pressure,massflow,temp,fraction)


PARAMETER_TYPES
/* name          unit */
   area          "m2"
   volume        "m3"
   ther_resist   "(m2 K)/W"
   heat_cap      "J/(kg K)"
   massflow      "kg/s"
   abs_pres      "N/m2"
   gas_const     "J/(kg K)"

CONSTANTS
/* name          value          unit */
   abs_zero      -273.15        "C"
   p_atm         101300         "N/m2"
```

The first two fields of a variable type declaration deserve no comment; the "kind" categorizes the variable as either a direction dependent flow type variable (e.g. mass flow, electrical current) or a direction dependent potential type variable (.e.g. temperature, pressure, voltage). Flow type variables, denoted THRU, should sum to zero when they are connected, potential type variables, denoted CROSS, are set equal when connected.

A link type declaration defines a set of variables with given types and in a prescribed order.

Parameter types are similar to variable types but lack the kind attribute.

4.3 Equations

The basic behavior of a continuous component is described by a set of scalar equations, written in a natural manner, defining relations between variables, time derivatives of variables, parameters, and constants. Conditional expressions and standard functions are allowed, as well as references to separately defined functions, written in regular programming languages, such as F77, C, or Pascal.

Ordinary differential equations and algebraic equations may be mixed freely and their order is irrelevant for the solution procedure. Time derivatives are specified by a prime symbol immediately following the variable name.

The number of equations is variable to allow finite difference models. Special constructs handle indexing, FOR to state a sequence of equations, SUM to add over indexed expressions. Index variables are integer valued and not declared separately. Variables, links, and parameters can be vector valued entities (see definitions below and example in wall model).

4.4 Assignment Modelling

Equation modelling is the rule in NMF, but models exist that require the use of assignments to model changes of distinct states. A typical example is a thermostat that has to remember its current state. Special variables called Assigned States are used for this modelling. A_Ss keep there values from the previous timestep until they are explicitly changed by assignments of the form

<assigned state> := <expression>

Assignments of this type are collected in a section ASSIGNMENTS, located after the EQUATIONS section. In each time step, the updates are performed after the solution of the equations.

A_Ss may not appear on links. If they have to be exported, that must be done via a regular variable, equated to the A_S.

4.5 Local Variables

Local variables can be declared in a model and be used as help variables to store values of expressions used repeatedly in the model. Their values are set by assignments, located in an ASSIGNMENT section in front of the equations. The values are not kept between evaluations of the model.

4.6 Links

All variables that connect a model to the outside world have to appear in link declarations in the model. The same variable may appear in more than one link declaration. A link may contain variables that are not used in the model equations, see e.g. pressure in the model 'bteem' below.

A link must either be of a globally declared link type, or else be declared GENERIC. In the normal case, connected links will be of the same declared type and compatibility of variable types will follow implicitly. The predefined link type GENERIC allows any number of variables of arbitrary types; it may be connected to a link of any type with the same number of variables, and connected variables will be type checked pairwise.

In the link declarations, each THRU variable is specified as to direction, POS_IN or POS_OUT.

Links may only contain scalar variables, i.e. single variables and indexed variables are allowed, but not entire variable vectors. Links may themselves be indexed, allowing a flexible number of ports for a model.

4.7 Variable Definitions

Variables come in two main groups, regular and assigned, each group further subdivided in two.

Regular variables occur only in equations and get their values by the global solution procedure.

Assigned variables are either help variables (LOC) or assigned states (A_S). They obtain their values locally in the component by assignments, from regular or other assigned variables.

Regular variables are cast to play a role in the definition, as either given (IN) or calculated (OUT). This corresponds to one well posed problem for the model; in most cases the roles are not fixed but may be reinterpreted by the solver in the context of a simulated system. The number of OUT variables must agree with the number of equations.

The variables are declared to be of a specified type, defined globally among variable types, or else GENERIC, matching any variable type. The type declarations are significant for variables appearing in links, but useful elsewhere for clarity.

Variable role has been defined above; for variables that only appear in links the choice is arbitrary, but conventionally IN.

For vector valued variables, variable name is declared with an index range within square brackets. The upper limit is defined by a model parameter (see next section).

Default, minimum, and maximum values appear as an optional group of specifications in a variable declaration. Min - max defines a validity range; default supplies an initial value for dynamic variables and assigned states, or an initial guess for algebraic variables. The defaults can be overridden when a simulation is started.

4.8 Parameter Definitions

A model definition describes a component type. In a simulated system, several module instances may be of the same component type, but will be diversified by different sets of parameter values.

Parameters are declared in the model under two headings: MODEL_PARAMETERS and PARAMETERS. Model parameters are used to adapt a model structurally; they take integer values. They are used to define sizes of vectors but may also be used as regular parameters in equations, usually as limits in FOR and SUM statements.

Parameters are typed globally in the same way as variables. Their declarations are also similar but lack the 'role' field. Parameters may be vectors but model parameters are always scalar integers.

4.9 Parameter Processing

Extra parameters may be declared in a model, in excess of those appearing explicitly in the equations. It is often convenient for the user to supply parameters in a setup differing from that appearing in the equations. These 'easy access parameters' will then be transformed to the internal ones by a subroutine specified under PARAMETER_PROCESSING. The routine may be defined locally or globally.

5 MODEL IMPLEMENTATION

The model library described in this paper is defined by NMF and implemented in IDA directly by a step for step translation. With other words, the NMF model description is complete, and the model could be ported directly to any simulation program capable for the problem.

This section presents the NMF definition of all the models used in the simulation. The definitions uses the global definitions already presented.

There are made some compromises to minimize the development time. Therefore some of the models are specialized to the problem we are solving. At a later stage they can be implemented more generally. The room model is for instance a bit un-symmetric and limited at the moment, but it fits well into our problem. It also demonstrates the flexibility of array variables.

5.1 Tee Piece for the Mixing of Two Flows

A tee piece for mixing two air flows includes heat balance, contaminant fraction balance and air mass balance, totally three algebraic relations. Note the notation of equations with the operator "=". The formulation of the left hand and right hand side expressions is free to the user.

```
CONTINOUS_MODEL bteem

ABSTRACT
"Mixing of two flows with no pressure loss. Specific heat capacity at
 both inlets is assumed to be equal and constant."

EQUATIONS
/* Mass conservation */
  M1 + M2 = M3;

/* Fume fraction conservation */
  X1*M1 + X2*M2 = X3*M3;

/* Energy conservation */
  M1*T1 + M2*T2 = M3*T3;


LINKS
/* type     name        variables... */
   p_m_t_x  inlet_1     P, POS_IN  M1, T1, X1
   p_m_t_x  inlet_2     P, POS_IN  M2, T2, X2
   p_m_t_x  outlet      P, POS_OUT M3, T3, X3

VARIABLES
/* type      name         role def  min        max descr */
   massflow  M1           IN   0    0          BIG "Massflow inlet 1"
   massflow  M2           IN   0    0          BIG "Massflow inlet 2"
   massflow  M3           OUT  0    0          BIG "Massflow outlet"
   temp      T1           IN   0    abs_zero   BIG "Temperature inlet 1"
   temp      T2           IN   0    abs_zero   BIG "Temperature inlet 2"
   temp      T3           OUT  0    abs_zero   BIG "Temperature outlet"
   rel_pres  P            IN   0    abs_zero   BIG "Pressure"
   fraction  X1           IN   0    0          BIG "Fume fraction inlet 1"
   fraction  X2           IN   0    0          BIG "Fume fraction inlet 2"
   fraction  X3           OUT  0    0          BIG "Fume fraction outlet"

END_MODEL
```

5.2 Tee Piece for the Splitting of Two Flows

A tee piece for splitting an air flow into two air flows includes mass balance, totally one algebraic equation.

```
CONTINOUS_MODEL btees

ABSTRACT
"Splitting of two flows with no pressure loss."

EQUATIONS
/* Mass conservation */
  M1 = M2 + M3;

LINKS
/* type     name        variables... */
   p_m_t_x  inlet       P, POS_IN  M1, T, X
   p_m_t_x  outlet_1    P, POS_OUT M2, T, X
   p_m_t_x  outlet_2    P, POS_OUT M3, T, X

VARIABLES
/* type      name         role def  min        max descr */
   massflow  M1           IN   0    0          BIG "Massflow inlet"
   massflow  M2           OUT  0    0          BIG "Massflow outlet 1"
   massflow  M3           OUT  0    0          BIG "Massflow outlet 2"
   temp      T            IN   0    abs_zero   BIG "Temperature"
   rel_pres  P            IN   0    abs_zero   BIG "Pressure"
   fraction  X            IN   0    0          BIG "Fume fraction"

END_MODEL
```

5.3 Fan with Polynomial Fan Curve

The fan is a simplest possible model using a fourth degree polynomial for the flow characteristic. So far, the fan is included in the model mostly to prove that the solver can handle the combination of two fans and the whole duct system. No model for dependence of air density on temperature is included at this stage.

```
CONTINOUS_MODEL bvft

ABSTRACT
"Simple fan model with 3. order polynomial fan characteristic and
 neglected fan work"

EQUATIONS
/* Fan characteristic */
   0 = -M + a + (P1-P2) * (b + (P1-P2) * (c + (P1-P2) * d));

LINKS
/* type     name        variables... */
   p_m_t_x  inlet       P1, POS_IN  M, T, X
   p_m_t_x  outlet      P2, POS_OUT  M, T, X

VARIABLES
/* type      name         role def  min        max descr */
   temp      T            IN   0    abs_zero   BIG "Fluid temperature"
   massflow  M            OUT  0    0          BIG "Massflow"
   rel_pres  P1           IN   0    0          BIG "Pressure at inlet"
   rel_pres  P2           IN   0    0          BIG "Pressure at outlet"
   fraction  X            IN   0    0          BIG "Fume fraction"
```

```
PARAMETERS
/* type        name      def     min    max descr */
   generic     a         0       -BIG   BIG "Coefficient 1"
   generic     b         0       -BIG   BIG "Coefficient 2"
   generic     c         0       -BIG   BIG "Coefficient 3"
   generic     d         0       -BIG   BIG "Coefficient 4"

END_MODEL
```

5.4 One Directional Pressure Resistance

The pressure resistance is a quadratic function of air volume flow and the pressure difference. The air volume is calculated by the ideal gas law. Therefore, the equivalent mass flow coefficient increases with the air temperature, an important effect in fire simulations.

This model uses a conditional expression to detect a point near zero where the quadratic pressure equation is replaced by a linear equation. The model can be been refined by letting the value of 1 Pa at the discontinuity be replaced by a parameter. The equation can easily be expanded to be valid for negative mass flow.

```
CONTINOUS_MODEL prst_1

ABSTRACT
"One way pressure resistance with linearized characteristic below a
 pressure difference of +1.0 Pa.  Air volume is modelled as a function
 of temperature and pressure."

EQUATIONS
/* Pressure equation, only valid for positive values of the pressure
   difference. */
  M*r*(T+273.15)/(P1+patm)) = kf * IF P1-P2>1.0 THEN
                                     SQRT(P1-P2);
                                   ELSE
                                     (P1-P2)
                                   ENDIF;
LINKS
/* type      name        variables... */
   p_m_t_x   inlet       P1, POS_IN  M, T, X
   p_m_t_x   outlet      P2, POS_OUT  M, T, X

VARIABLES
/* type       name          role def  min         max descr */
   massflow   M             OUT  0    0           BIG "Massflow"
   rel_pres   P1            IN   0    0           BIG "Pressure at inlet"
   rel_pres   P2            IN   0    0           BIG "Pressure at outlet"
   temp       T             IN   0    abs_zero    BIG "Bulk temperature"
   fraction   X             IN   0    0           1   "Mass fraction of fume"

PARAMETERS
/* type         name      def    min   max descr */
   massflow     kf        SMALL  SMALL BIG "Flow coefficient"
   gas_const    r         SMALL  SMALL BIG "Gas constant"
   abs_pres     patm      p_atm  0     BIG "Atmospheric pressure"

END_MODEL
```

5.5 One Directional Pressure Resistance with Thermal Heat Loss.

The duct is modelled as a tank with heat loss combined by a pressure resistance at the outlet. This version of the model provides no discretization into smaller air volumes. Discretization of the air volume can be achieved also with this model by coupling several components in a series.

Environmental heat transfer is included in the model by a constant heat resistance to reduce the total number of modules in the simulation model.

No thermal air mass or duct wall mass is taken into account. A later need for modelling wall mass can be solved by connecting the "wall" link to a WALL model, letting the thermal resistance parameter represent the inside convective heat resistance.

```
CONTINOUS_MODEL prst_3

ABSTRACT
"Ventilation duct with one single air temperature and a pressure
 resistance at outlet. Heat loss through the duct wall to ambient
 temperature.  One way pressure resistance with linearized
 characteristic below a pressure difference of +1.0 Pa.  Air volume is
 modelled as a function of temperature and pressure."

EQUATIONS
/* Air temperature */
  0 = -M*cp*(T-Ti)
     -avegg/rvegg*(T-Ta);

/* Heat flux from ambient */
  Qa = avegg/rvegg*(Ta-T);

/* Pressure equation, only valid for positive values of the pressure
   difference. */
  M*r*(T+273.15)/(P1+patm)) = kf * IF P1-P2>1.0 THEN
                                     SQRT(P1-P2);
                                   ELSE
                                     (P1-P2)
                                   ENDIF;
LINKS
/* type      name        variables... */
   t_q       wall        Ta, POS_IN Qa
   p_m_t_x   inlet       P1, POS_IN  M, TI, X
   p_m_t_x   outlet      P2, POS_OUT  M, T, X

VARIABLES
/* type       name        role def  min        max descr */
   temp       T           OUT  0    abs_zero   BIG "Bulk temperature"
   heatflux   Qa          OUT  0    0          BIG "Amb. heat flow"
   massflow   M           OUT  0    0          BIG "Massflow"
   rel_pres   P1          IN   0    0          BIG "Inlet pressure"
   rel_pres   P2          IN   0    0          BIG "Outlet pressure"
   temp       Ti          IN   0    abs_zero   BIG "Inlet temperature"
   temp       Ta          IN   0    abs_zero   BIG "Ambient temperature"
   fraction   X           IN   0    0          1   "Fume fraction"

PARAMETERS
/* type          name      def    min   max descr */
   massflow      kf        SMALL  SMALL BIG "Flow coefficient"
   gas_const     r         SMALL  SMALL BIG "Gas constant"
   abs_pres      patm      p_atm  0     BIG "Atmospheric pressure"
   heat_cap      cp        SMALL  SMALL BIG "Spes. heat capacity of air"
   area          avegg     SMALL  SMALL BIG "Wall surface area"
   ther_resist   rvegg     SMALL  SMALL BIG "Heat resistance pr m2 of wall"

END_MODEL
```

5.6 Wall

The wall is modelled as series of thermal resistances and capacitances with values given manually. This require manual discretization of the of the wall. Parameter processing could have saved the user a lot ov work by allowing the user specify the thermal values and thickness of each layer of the wall, but at this stage of the project this is not applied to the model.

On the other hand, the model is flexible. In our case with no separate radiation model, the convective and radiative heat resistance is included as the first and last resistance.

The model uses arrays with user specified size given by the model parameter **n**. In this case, both variables and parameters are indexed. The DO loop operate on all segments of the wall.

Special care is taken at the boundaries, and separate boundary variables are defined. The boundary variables are necessary in this because a single array element can not be a part of a link. However, whole arrays can be a part of link, as demonstrated in the room model.

Note the occurrence of T as a prime variable (T'[i]) in the equations for the internal nodes.

```
CONTINOUS_MODEL wall

ABSTRACT
"Wall with variable number of user specified thermal resistances and
 heat capacities"

EQUATIONS
/* Heat balance for all the internal nodes */
  FOR i=1,n {
    c[i]*a[i]*T'[i] = -(T[i] - IF i=1 THEN
                                 T1
                               ELSE
                                 T[i-1]
                               ENDIF)  * a / IF i=1 THEN
                                               r0
                                             ELSE
                                               r[i-1]
                                             ENDIF
                      -(T[i] - IF i=n THEN
                                 T2
                               ELSE
                                 T[i+1]
                               ENDIF) * a/r[i];
  }

/* Heat balance for surface 1 */
  0 = Q1 - (T1-T[1])*a/r0;

/* Heat balance for surface 2 */
  0 = Q2 - (T2-T[n])*a/R[n];

LINKS
/* type      name       variables... */
   t_q       side_1     T1,POS_IN Q1
   t_q       side_2     T2,POS_IN Q2

VARIABLES
/* type       name        role def  min       max descr */
   temp       T1          OUT  0    abs_zero  BIG "Side 1 temperature"
   temp       T2          OUT  0    abs_zero  BIG "Side 2 temperature"
   temp       T[    n]    OUT  0    abs_zero  BIG "Internal temperatures"
   heatflux   Q1          IN   0    -BIG      BIG "Side 1 heat flow"
   heatflux   Q2          IN   0    -BIG      BIG "Side 2 heat flow"

MODEL_PARAMETERS
/* type         name   min max      descr */
INT             n      1   BIGINT "Number of internal nodes"

PARAMETERS
/* type          name      def    min   max descr */
   area          a         SMALL  SMALL BIG "Wall area"
   ther_resist r0          SMALL  SMALL BIG "First thermal resistance/m2"
   heat_cap    c[    n]    SMALL  SMALL BIG "Heat capacity/m2"
   ther_resist r[    n]    SMALL  SMALL BIG "Thermal resistance/m2"

END_MODEL
```

5.7 Room Model

The room is modelled as simple as possible in this first approach. One single temperature is calculated.

It should be noticed that no external heat transfer is included in the room model itself, neither convective nor radiative. Variable number of interfaces for air inlet, air outlet and heat load is used instead. In our system the convection model is connected to the wall model by simple resistances. No radiation model is used, but could also be added to the system by a separate component. The model library is therefore supported by a great flexibility and minimizes the need of rewriting the model later.

Air leaks to other rooms are modelled by one-directional air inlets and air outlets. It is possible to define interfaces for two directional air flow, but it this is not done in this early approach. Instead, we must become sure that air is flowing the right direction by carefully select the pressure resistances of the system.

Another approach to calculating air leaks is assuming the leak resistances to be small in comparison to the ductwork resistances. This is sometimes the case in real buildings. We have made a simulation of this kind by connecting the leak interfaces directly without using pressure resistances like in Figur 1 with good result.

Air density is modelled by the ideal gas law, and this will shorten the time constant of the room air temperature as temperature increases. Note that total air mass is defined as a local variable and given its value in an assignment with the operator ":=" in an ASSIGNMENTS section **before** the EQUATIONS section.

Smoke and air mixing is modelled by a separate differential equation assuming total air mixing.

This model uses three model parameters enabling three array groups to exist independently. The model has no internal discretization as the wall model. Instead the arrays are acting in interfaces with variable numbers.

```
CONTINOUS_MODEL nroom

ABSTRACT
"Room model with thermal mass in air and a variable number of heat
 losses. The room has a variable number of air inlets and air
 outlets.  Air volume is modelled as a function of temperature and
 pressure. A fume gas fraction is calculated."

ASSIGNMENTS
/* Room air mass is handled as an assigned state */
  Mair := (P+patm)*V/(r*(T+273.15));

EQUATIONS
/* 1: Room air heat balance */
  0 = -Mair*cp*T'
      -SUM i=1,nin Mi[i]*cp*(T-Ti[i])
      +SUM i=1,nq Q[i];

/* 2: Room fraction balance */
  0 = -Mair*X'
      -SUM i=1,nin Mi[i]*(X-Xi[i]);

/* 3: Room air mass balance */
  0 = -M
      -SUM i=1,nout Mo[i]
      +SUM i=1,nin Mi[i];

LINKS
/* type     name        variables... */
   p_m_t_x  inlet       P, POS_IN Mi, Ti, X
   p_m_t_x  main_outlet P, POS_OUT M, T, X
   p_m_t_x  outlet      P, POS_OUT Mo, T, X
   t_q      surfaces    T, POS_IN Q
```

```
VARIABLES
/* type      name          role def  min       max descr */
   temp      T             OUT  0    abs_zero  BIG "Outlet temperature"
   fraction  X             OUT  0    0         BIG "Fume fraction"
   massflow  M             OUT  0    0         BIG "Main outlet massflow"
   massflow  Mi[   nin]    IN   0    0         BIG "Inlet massflow"
   massflow  Mo[   nout]   IN   0    0         BIG "Extra outlet massflow"
   rel_pres  P             IN   0    0         BIG "Pressure"
   heatflux  Q[   nq]      IN   0    0         BIG "Surface heat flow"
   temp      Ti[   nin]    IN   0    abs_zero  BIG "Inlet temperature"
   temp      Xi[   nin]    IN   0    0         BIG "Inlet fume fraction"
   mass      Mair          LOC  0    0         BIG "Room air mass"

MODEL_PARAMETERS
/* type       name   min max    descr */
   INT        nin    1   BIGINT "Number of air inlets"
   INT        nout   0   BIGINT "Number of additional air outlets"
   INT        nq     1   BIGINT "Number of wall interfaces"

PARAMETERS
/* type        name     def    min   max descr */
   heat_cap    cp       0      0     BIG "Sp. heat capacity of air"
   volume      v        0      0     BIG "Room volume"
   ther_resist r        SMALL  SMALL BIG "Gas constant"
   abs_pres    patm     patm   0     BIG "Atmospheric pressure"

END_MODEL
```

6 DISCUSSION AND CONCLUSIONS

The presented example is a practical demonstration where the NMF syntax is followed in detail for model description. The problem demonstrates the major features of NMF. How well does NMF actually work as a tool for model development? Let us assume that automatic translation is available.

Our major observation is the low threshold for understanding the whole simulation without knowledge about the application environment (IDA). Several people have been involved in the project, and the NMF has been the only model description used. All problems occurring in the simulations have been solved by studying the NMF descriptions (except some errors caused by the manual translation).

An important feature is portability. Exchange of NMF files can be a convenient way of porting files between quite different simulation programs.

We have not tried the feasibility of NMF for model specification to for example TRNSYS, but part from the LINKS feature, which is irrelevant for this environment, the format should be applicable. The input/output modelling of TRNSYS can be handled by adding restrictions to how the equations are written. An even more flexible approach would be letting the automatic translator convert the equations into assignments using symbol manipulation.

The present version of NMF lacks a hierarchy which allows several versions of a model share the common model kernel. This is demanded when writing practical applications. The initial idea is letting the simulation environment take care of this part of the problem. NMF will as it is defined today, serve as a format for documentation and porting of specific, ready-to-run models. At a later stage, hierarchical components can be discussed, making the NMF more complete.

The NMF neutral model format has proved to be a promising tool for model development. Its simple and open syntax is suitable for describing most common models that appear in building system simulation. When automatic translators become available (they are under development for the IDA system), it will become a powerful tool. The format is worth further exploration in connection to other simulation environments. With the neccessary alterations, NMF could be a good platform for a common future model definition standard.

The NMF neutral model format has proved to be a promising tool for model development. Its simple and open syntax is suitable for describing most common models that appear in building system simulation. When automatic translators become available (they are under development for the IDA system), it will become a powerful tool. The format is worth further exploration in connection to other simulation environments. With the neccessary alterations, NMF could be a good platform for a common future model definition standard.

REFERENCES

Sahlin, P. and Sowell, E.F. 1989. "A Neutral Format for Building Simulation Models". Proceedings of the Building Simulation '89 Conference, Vancouver, Canada, June 1989.

Bring, A. 1990. "IDA SOLVER, A Users Guide". Dept. of building Services Engineering, Royal Institute of Technology, Stockholm, Sweeden.

Bring A. and Sahlin, P. 1991. "IDA SOLVER, A TOOL FOR BUILDING AND ENERGY SYSTEMS SIMULATION. Proceedings of the Building Simulation '91 conference, Nice, France.

Kolsaker K. 1990 "DYMAMISK SIMULERING MED IDA. Et praktisk verktøy for bygningssimulering med modellbibliotek for fjernvarmeinstallasjoner". Technical report ISBN 82-595-6091-7, SINTEF, Trondheim, Norway.