PHYSICAL SYSTEM MODELLING LANGUAGES : FROM ALLAN TO MODELICA

Alexandre JEANDEL, Fabrice BOUDAUD
GAZ DE FRANCE - Research and Development Division
BP 33 - 361 avenue du Président Wilson
F- 93211 SAINT DENIS La Plaine CEDEX - FRANCE
alexandre.jeandel@edfgdf.fr

ABSTRACT

Modelling and simulation play a number of roles in engineering design studies. For Gaz de France, these studies must satisfy exacting criteria of quality and rapidity. Studies are even more effective if models developed on previous occasions can be stored and reused and if the company is able to share models with its partners. The development of model exchanges is therefore a key factor determining the future scope of modelling/simulation activities.

The Modelica design group was set up to design of a new language for physical modelling. There are already several modelling tools with their own language for object-oriented, non-causal modelling, such as Dymola, gPROMS, MOSES, NMF, Omola, ALLAN and U.L.M. This working group is attempting to unify the concepts and to introduce a common basic syntax and semantics.

Gaz de France is very much looking forward to the existence of such a language, shared by different users with different modelling environments. We will present a simlifide model of a water heater in ALLAN and in Modelica.

1. INTRODUCTION

For many years, Gaz de France has been using modelling/simulation techniques to improve the design of complex technical systems such as boilers, air-conditioning systems, industrial furnaces and gas distribution networks.

Modelling and simulation play a number of roles in our studies. They provide us with a clearer understanding of our installations, producing rich and valuable information about the way a real technical system actually works. Simulation also enables us to design experimental protocols and even to make extrapolations of behaviour in domains that cannot easily be tested in the laboratory. Lastly, and above all, technical system design may involve the optimization of certain components or their replacement by different ones.

For Gaz de France, these studies must satisfy exacting criteria of quality and rapidity. Studies are even more effective if models developed on previous occasions can be stored and reused and if the company is able to share models with its partners. The development of model exchanges is therefore a key factor determining the future scope of modelling/simulation activities.

First, we will examine our initial attempts to improve these exchanges and our relatively long experience in the field of modelling languages. We will then briefly recall the history of model languages at Gaz de France : ALLAN and ULM. An outline of the necessary characteristics of a modelling language will follow. Gaz de France is now taking part in the European attempt to design a new physical system modelling language : "Modelica". The same simplified model of a boiler will be presented in the ALLAN and in the Modelica language.

2. MODELLING LANGUAGES

2.1. ALLAN NEUTRAL LANGUAGE

Gaz de France has acquired extensive experience of a neutral modelling language through the use of the ALLAN.™ software. Indeed, by 1983, the Gaz de France Research and Development Division had already designed a general description and simulation program for dynamic systems which was developed in association with CISI [Pottier 1983, Jeandel 1997]. It is able to perform simulation using one of two solvers : NEPTUNIX or ADASSL (a customized version of DASSLRT) [Petzold 1982].

Though close to NEPTUNIX, ALLAN original language was intended to be totally independent of the first two solvers used: NEPTUNIX and ASTEC [Nakhlé 1991]. The latest version of this language is used since version 3.0 of ALLAN. It has a physical system design engineer oriented language. It moves away from computer science and numerical analysis. It has some important features such as:

1. Use of physical entities (physical variables, parameters, etc.)
2. Use of signal entities (signal variables, logical variables, etc.)
3. Declarative description of continuous behavior

™ ALLAN. Is a Gaz de France trademark

(non-oriented)
4. Ability to describe conditions, discontinuities and events
5. Use of an automaton to describe synchronous actions
6. Ability to call FORTRAN or C subroutines or functions.
7. Independent of solver

Strong semantics are linked to the features and over the last 15 years they have proved to be very useful. Some features are covered by the ALLAN software but not by the textual language.

1. It is limited to the description of a simple model and cannot be used to describe a system,
2. It mixes numerical data with the equational structure of the model,
3. It does not take account of semantics associated with compound models and operation,
4. It does not take account of the modelling work procedure and, in particular, of validation,
5. And above all, only ALLAN is able to process these models.

A tool-dependent language is isolating and does not permit model exchange with partners. For this last reason, in 1993, Gaz de France turned its attention towards the Neutral Model Format "NMF".

2.2. "NMF" AND ULM

We felt that the work being carried out on an international level and within ASHRAE for the standardization of a modelling language was moving in the right direction and deserved to be supported [Sahlin & al. 1989 to 1996]. The characteristics of the NMF are similar to those of the initial ALLAN and were therefore, in principle, compatible with our needs. Unfortunately, we soon discovered that this description language also had its limits and decided to demonstrate our need by developing a new language.

Our aim was to obtain a general pivot language, both neutral and user-friendly. We called it ULM, standing for "Un Langage de Modélisation" which means : "a modelling language".

The feasibility of ULM has been tested on the ULM to ALLAN link [Jeandel & al. 1995]. The language and the corresponding translator were developed in 80 days in 1993. The translator is only available on an old version of ALLAN. In our work context, ULM could be used as a pivot language between ALLAN and other software to permit the exchange of models with outside users.

Through ULM, Gaz de France has presented its modelling language needs to the scientific community and to modellers. For us, ULM constitutes the minimum that a language must provide in order to be of interest to us. We have even taken things further by demonstrating the feasibility of a specific solution and proposing such a solution. However, our real aim is to encourage modellers to adopt one or more modelling languages which are compatible with each other. The initiative of Hiding Elmquist in 1996 provided an answer to our quest.

2.3. MODELICA

First, as part of the ESPRIT project "Simulation in Europe Basic Research Working Group (SiE-WG)", a group was set up to design of a new language for physical modelling. There are already several modelling tools with their own language for object-oriented, non-causal modelling, such as Dymola, gPROMS, MOSES, NMF, Omola, ALLAN and U.L.M. This working group is attempting to unify the concepts and to introduce a common basic syntax and semantics. It has now become the first technical committee within Eurosim (the federation of European Simulation Societies). The members of the Modelica design group are listed at the end of this paper.

The work started in the continuous time domain since there is a common mathematical framework in the form of differential-algebraic equations and there are several existing modelling languages based on similar ideas. There is also significant experience in the use of these languages in various applications. It thus seemed to be appropriate to collect all knowledge and experience and to design a new unified modelling language or neutral format for model representation. Thus the short-term goal is to design a modelling language for differential-algebraic equation (DAE) systems with some discrete event features to handle discontinuities and sampled systems. This design is in progress. It should be extendable so that the goal can be expanded to the design of a multi-formalism, multi-domain, general-purpose modelling language.

Gaz de France is very much looking forward to the existence of such a language, shared by different users with different modelling environments. Before we present a model in ALLAN and in Modelica, let us summarize Gaz de France's requirements for such a language.

3. REQUIREMENTS FOR A PHYSICAL SYSTEM MODELLING LANGUAGE

3.1 LANGUAGE CHARACTERISTICS

The very first requirement is that the language must enable us to do as well as, or better than today in our design studies involving simulation of technical systems (continuous behavior and events ; independence from computer science and numerical analysis). But beyond that, it needs to be a general pivot language, both neutral and user-friendly. This means that we should be able to exchange models between environments without losing information

and that the descriptions should remain independent of the environment.

The main difficulty is to make the compromise between a very general language, capable of describing all the models, and a language with the right entities for the user and strong semantics to help him. It is a compromise between generality and specificity.

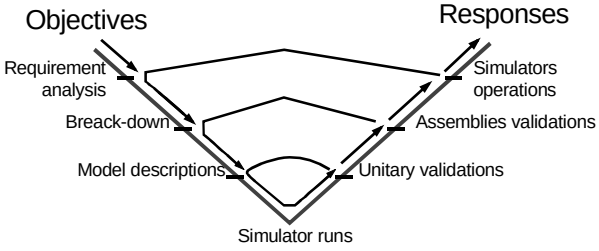
The specification of a language is the result of an analysis of the nature of modelling/simulation. In the following text, we will underline, both on the paper and in the figurative sense, the elements that we believe to be essential to the definition of the language.

3.2 IMPACT OF STUDY OBJECTIVES

What do we do ? We assess our knowledge of a given technical system or extrapolate its behaviour under new conditions. We also design systems or optimize their components. For experimental planning, we use simulation to optimize the sollicitations needed for a given experiment on our test benches. As a control department we synthesize and/or evaluate control or diagnosis algorithms on simulators. All these study objectives mean that we have to be able to deal with certain entities of the simulator as such. Let us name : excitations, commands, observations, system structure, numerical values of system parameters, evolutive variables of the system and their initial values. It should be noted that the notion of observation of a technical system is different from that of the output of this system. The latter being more a question of relationships between components or a computer science idea. And, in order to synthesize commands, the language must permit the coupling of dynamic continuous behaviour with the numerical command algorithm. It must be possible to handle the problems of coincidence between the continuous domain and the discrete domain linked to numerical controller.

However, to be usable, a language must take account not only of what needs to be handled but also how this is achieved.

3.2 IMPACT OF THE STUDY APPROACH

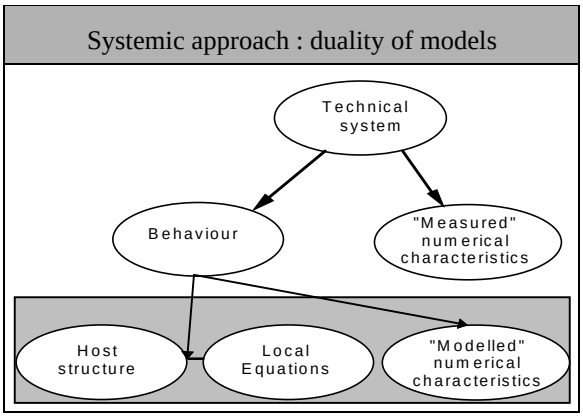


For several years now, our work has followed a standard quality approach. The study process progresses from the analysis of the problem to its answer via a number of stages and validations. It has the shape of an iterative V-cycle. Particular attention is paid to the definition of needs, to the role of each person involved and to the various validation phases.

Ideally, a modelling language should take account of each of these steps. Even though it is possible to represent this process in a linear manner, all modellers know that they are constantly coming and going between the various stages. The modelling language must therefore take account of this need for iteration during the modelling process. It is not an easy task, as it means that the language must be able to keep track of model versions and model uses. It must be possible to repeat what has already been done.

A study can be carried out without any modelling, assuming that the models are available in a model library. You first need to have the right model for a given question. But, even then you must have complete transparency and full information in order to have confidence in your model. This means that the modelling language should be very easy to read, that documentation should be available and that you should be able to re-run the validation cases.

The validation phase is rarely formalized, though it can be in certain cases. Indeed, it must be so for reasons of scientific rigour. A study of the nature of validation [Jeandel 1995, Izquierdo & al. 1995] has shown that it is possible to provide language elements specific to the validation phase: simulations, reference values, comparison criteria.



Most of the studies are based on model simulation, some work directly on the model as such. This latter case will be taken into account in the analysis of the modelling approach. For simulation you have, on one side, the set of equations describing the behaviour of your system, and you look for the right numerical data to put in. An object approach to the model clearly shows us that we have both the object comprising the model equations (host structure + local equations) and its instantiation: the numerical data. This model duality should be found in the language. It is reproduced in the figure above.

3.3 MODELLING APPROACH IMPACT

The practical approach to the modelling of complex system is based on a breakdown of the system into sub-systems. In practice, structured models can be built from bottom up (by assembly of simple models) or from top down (by breakdown). It must be possible to break a model down again into other models at any time. Similarly, it must be possible to use the model with other models to form a new one in an equally flexible manner. The language must be very flexible with respect to the model hierarchy and its pathway.

The models are thus contained in a host structure which ranges from simple assignment relations between isolated variables (oriented block diagrams), to the generation of all sorts of equations, depending on the method used, and to the *Bond Graph*. The model host structure is not a property specific to models declared as "complex", but must be considered as general framework. It must be possible to use the language to make a formal definition of relations between models.

3.4 REPRESENTATION LEVEL IMPACT

People work at different representation levels. One group of users may only be interested in dealing with ready-to-run simulators. Some other users will only deal with technical systems represented by icons. You will find users only speaking in term of physics. You may also encounter people only interested in the mathematical model or even in the computer code. Others are only interested in the mathematical model and it is difficult to define at what level people will want to exchange models. It would be very nice if model exchanges could take place at these different levels. U.L.M. for example, is able to sort these different levels and exchange the right information.

Let us say a few words about a very special representation level. For a model to be reusable it must be as general as possible. There is a science whose purpose is to provide a general and consistent framework for these models, i.e., physics. A language cannot ignore physics. It is essential to respect the principles of physics. Therefore it is necessary, for example, to handle the dimensions of the variables, their ranges of values and automated abilities for conservation-law balances.

3.5 IMPACT OF THE NUMERICAL SOLVER

From the physical point of view, there are "primitive" variables (characterizing the initial physical model) and "secondary variables" (which can be expressed as a function of primitive variables and parameters and which may or may not require specific physical interpretation). This feature could be included in the language and be used to simplify the work of the solver.

3.6 IMPACT OF THE COMPILER

The language must be used with a computer and must be defined in association with the development of a corresponding translation software. The language must be defined with its semantic and contextual controls. A language should only include elements that can be verified. In fact, this is not completely compatible with the documentation requirements and compromises need to be made.

It is possible to identify three different types of language elements. Some of them are required to run the models, like the equations in the models, the equations generated from the links between the models, and the numerical data values. Others can be used for consistency checking, like quantities, dimensions etc.. And a third type of information is only there for documentation, like the diagram associated with a model or the string label linked to an entity. For the string label, you are supposed to give a definition of the entity but there is no way to check what is actually written. One way is to propose a list in the modelling environment and to allow only one choice within this list. This is again very difficult because of the very different needs of the different users.

Then, provided it does not affect language user-friendliness, the language must be well suited to the world of computers. It must therefore respect simple grammatical rules of the language theory. This will facilitate its widespread use. It would be a pity if the language was designed in such a way that subsequent development of software using the language was a long and costly business.

3.7 SOFTWARE IMPACT

However, the establishment of links with computer programs brings to light certain constraints due either to the choice of representation of the software concerned or due simply to the type of computers used. A language and, above all, the associated translator, must be capable of handling computer constraints, such as the length of the names used, or computer zeros and infinites. It may seem quite ridiculous, but what can you do if the names of your ten thousand variables are eight characters long and your simulation software only allows six ? These constraints must be taken into account in the translator software specification so that the use of the language is made quite transparent to the modeller.

4. THE MODEL OF A WATER HEATER

4.1 PRESENTATION

This is a model from the initial ALLAN library. The water heater was broken down into several components :

1. a burner (11- 32 kW limited operating range),

2. a heat exchanger,
3. a jacket,
4. a water feed pipe,
5. a water flow pipe,
6. a temperature sensor,
7. a gas inlet.

The modelling was straightforward, based on the following assumptions :

The heat-exchanger is likened to a one-dimensional pipe discretized into 5 sections. It takes in cold water represented by temperature-mass flowrate. It receives heat from the burner and delivers hot water.

The inside of the water-heater is delimited by a jacket at a uniform temperature. The exchanges between the jacket of the bathwater heater and the environment occur at a uniform temperature.

The burner operates in on-off mode for gas input. It emits a radiant output and a convective output. The distribution of those two outputs was identified on an experimental test bench. The ignition and extinction fronts are represented by first order equations.

The draw-off pipe outside the jacket is represented by a simple model of a temperature-mass flowrate pipe.

The temperature sensor is assumed to be perfect.

The gas inlet to the burner is represented by a heat input. The n.c.v. of the gas is therefore assumed to be constant.

4.2. SIMPLE MODEL TEXT

As an example, below is the model of the burner in the two representations.

Since this model is completely oriented, it is called BLOCK in the Modelica text. For the same purpose ALLAN. makes use of keyword ENTREE (input) and SORTIE (output). But it is somewhat different, since you can mix oriented and non-oriented variables. Both languages allow if-then-else statements. The Modelica statement could be written repeating der(P) in such a way that it would look exactly the same as in ALLAN. We notice that the Modelica declaration part is somewhat shorter. This is due to the fact that the initialisations of the coupling variables take place in the heading and that some of the characteristics of the variables are pre-defined using an OO feature: the type declaration of

ALLAN.™

```
MODELE BRULE
 (ENTREE PNOM  (Pnom   : "VCO W");          -- Commande
  ENTREE QINFO (Qinfo  : "VCO kg/s EAU");  -- Debit d'eau
  SORTIE FUM   (Fum    : "VCO W");         -- Puissance fumees
  SORTIE RAY   (Ray    : "VCO W"))         -- Rayonnement

DECLARATIONS
    VAR INDEP   :    T unite="s";              -- Time
    PARAM :          Q1 = 0.06 "kg/s",         -- Water flow
                     Q2 = 0.04 "kg/s" ,        -- Water flow
                     Tb = 1 "s",               -- Time const.
                     Ds = 0 "kg/s2" ,          --Mini.  flow
                     Alf = 0.5  unite="" ;          --    Repart.
coef.
    VAR CONTINUE  : P   = 0 "W" ;
    VAR COUPLAGE :                             PNOM  = 0 ,
                     QINFO = 0 ,
                     FUM   = 0 ,
                      RAY   = 0;
    BOOLEEN :        DQINFO     vraisi    QINFO' < DS,
                     QMIN       vraisi    QINFO  < Q1,
                     QMED       vraisi    QINFO  < Q2;
EQUATIONS
  SI Dquinfo ALORS
    SI Qmed ALORS
      P' = -P/TB;
    SINON
      P' = (Pnom-P)/Tb;
  SINON
    SI Qmin ALORS
      P' = 0;
    SINON
      P' = (Pnom-P)/Tb;

  Ray = Alf*P;
  Fum = (1 - Alf)*P;
FIN
```

Modelica

```
BLOCK Brule(
    Entree : Pnom (start=0, Exact=FALSE),
             Qinfo (start=0, Exact=FALSE);
    Sortie :  Fum (start=0, Exact=FALSE),
             Ray (start=0, Exact=FALSE);



PARAMETER Mass_flow :    Q1=0.06,
                         Q2=0.04 ;
PARAMETER Time :         Tb=1 ;
PARAMETER Real :         Ds(Unit="kg/s2", Value=0) ;
PARAMETER Real :         Alf(Unit="")=0.5 ;
PARAMETER Thermal_flux : P(start=0, Exact=FALSE);


PARAMETER Boolean :      Qinfo  = der(Qinfo)<Ds,
                         Qmin  = Qinfo<Q1,
                         Qmed = Qinfo<Q2;


EQUATION

   der(P) =
        IF Dqinfo THEN
              IF Qmed THEN
                    -P/Tb
              ELSE
                    (Pnom-P)/Tb
        ELSEIF Qmin THEN
                    0
        ELSE
                    (Pnom-P)/Tb;

    Ray = Alf*P;
    Fum = (1-Alf)*P;
)
```

Modelica. We choose here to predefine physical variables according to their quantities. This way of proceeding places major constraints on the modeller and has to be validated experimentally.

Here is an example of the Modelica type definitions :

```
TYPE Real(
    EXTENDS VariableAttributes; RealType: value;
    PARAMETER StringType: Quantity = "";
    PARAMETER StringType: Unit = "";
    PARAMETER StringType: DisplayUnit = "";
    PARAMETER Alternative (Equal, Sum) : Connection =
    Equal;
    PARAMETER RealType: Min=-Inf, Max=+Inf;
    PARAMETER RealType: Magnitude=1;
    PARAMETER RealType : Start=0;
    PARAMETER BooleanType : Exact=TRUE;
    EQUATION
   ASSERT(value >= Min AND value <= Max, "out of limit");  )
TYPE Boolean(
    EXTENDS VariableAttributes ;
    BooleanType : Value;  )
TYPE Entree=Real(Unit="", Causality=Input)
TYPE Sortie=Real(Causality=Output)
TYPE Temperature=Real(Unit="K", DisplayUnit="Deg-C",
    Min=-273.15, Quantity="TEMPERATURE")
TYPE Thermal_flux=Real( Unit="W",
    Quantity="FLUX_THERMIQUE",  Connection=Sum)
    type Mass_flow=Real(Unit="kg/s",
    Quantity="DEBIT_MASSE", Connection=Sum)
TYPE Heat_value=
    Real(Unit="J/kg/K",Quantity="CHALEUR_MASSIQUE")
TYPE Mass=Real(Unit="kg", Quantity="MASSE")
TYPE Time=Real(Unit="s", Quantity="TEMPS")
CONNECTOR Tq( Temperature : T(Start=20, exact =false);
    Mass_flow : Q (Start = 0, exact = false))
```

In fact, the features described here in Modelica are implicit to the ALLAN. software. As such Modelica offers more possibilities, though model management is more difficult. When models are exchanged, it is important to provide the complete set of definitions needed to allow the model to run.

5.3. ITERATED SIMPLE MODEL

In Modelica, the use of FOR loops can be very useful for multiplying a model. This is a very simple heat exchanger model.

The simple models are again very much alike in the two languages. We notice the use of a PARAMETRE GLOBAL in ALLAN that is common to all models. The heat value of a specific material is the same everywhere and it is useful to insure consistency throughout the models. In Modelica, it should be possible to predefine an element in a similar manner. This implies the need to manage all default entities in the software. The textual language of ALLAN does not have For loops, nor arrays, the only possible way of doing the same thing in ALLAN is to define it graphically. The corresponding Modelica text could of course be generated. The INITIALISATION block in ALLAN is used to define models of the initial value, in this case the same thing can be done in the declaration of Modelica. We can also notice that Modelica makes use of the dot notation for the coupling variables.

ALLAN.™

```
MODELE CC2DX(
    TENC (TENC   : "%C") ;        -- Temperature enceinte
    ENTREE  RAY  (RAY    : "VCO W") ;   -- Rayonnement
    TQE  (TE     : "%C EAU",      -- Temp. d'entree
          QE     : "kg/s EAU") ;  -- debit d'entree
    TQS  (TS     : "%C EAU",      -- Temp. de sortie
          QS     : "kg/s EAU") ;  -- debit de sortie
    PERT (PERT   : "W"))          -- Puissance perdue
DECLARATIONS
  VAR INDEP      T unite="s";     -- Temps
  PARAM  :
     M  = 1.2 "kg" , MA = 0.16 "kg" ,  F  = 0.1 "" ,  N  = 5 "" ,
     H  = 138.7 "J/%C/s" , T1 = 5.37 "s" ,  T2 = 48.335 "s" ,
     T3 = 40.28 "s" ;
  PARAM GLOBAL  :
     CP unite="J/kg/%C";
  VAR CONTINUE :
     TA = 20 "%C" ;
  VAR COUPLAGE :
     TE   = 20 , QE   = 0 , TENC  = 0 , RAY   = 0 ,
     TS   = 20 , QS   = 0 , PERT  = 0 ;
INITIALISATIONS
  T1   = MA*CP/H/(1 - F);
  T2   = MA*CP/H/F;
  T3   = M*CP/H/(1 - F);
EQUATIONS
  0 = QE + QS;
  TA'  = (TS - TA)/T1 + (TENC - TA)/T2 + RAY/MA/CP;
  TS'  = (TA - TS)/T3 - (TS - TE)*(N*QE)/M;
  PERT = H*(TENC - TA)/N;
FIN
```

Modelica

```
MODEL Cc2dx(

    Tq : Tqe, Tqs;
    Temperature : Tenc(start=0, Exact=FALSE);
    Entree : Ray(start=0, Exact=FALSE);
    Thermal_flux : Pert(start=0, Exact=FALSE);




    PARAMETER Mass : M=1.2, Ma=0.16;

    PARAMETER Real : F=0.1, H(Unit="J/K/s",Value=138.7);

    PARAMETER Heat_value : Cp;

    PARAMETER Temperature : TA(start=20, Exact=FALSE);

    PARAMETER Real : N=5;


    PARAMETER Time :      T1(Value=Ma*Cp/H/(1-F)),
                          T2(Value=Ma*Cp/H/F),
                          T3(Value=M*Cp/H/(1-F));

EQUATION
    Tqs.Q = -Tqe.Q;
    der(Ta) = (Tqs.T-Ta)/T1+(Tenc-Ta)/T2+Ray/Ma/Cp;
    der(Tqs.T) = (Ta-Tqs.T)/T3-(Tqs.T-Tqe.T)*(N*Tqe.Q)/M;
    Pert = H*(Tenc-Ta)/N;
)
```

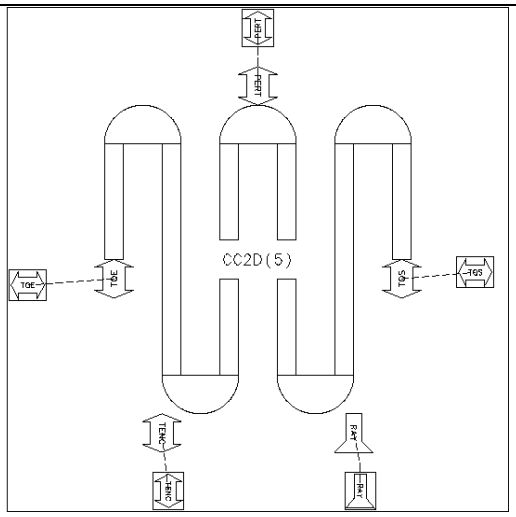


Figure 1 : External diagram of the iterated heat exchanger in ALLAN

```
MODEL Cc2d5(
    Tq : Tqe, Tqs;
    Temperature : Tenc(start=0, Exact=FALSE);
    Real : Ray(start=0, Exact=FALSE);
    Thermal_flux : Pert(start=0, Exact=FALSE);
    Cc2dx (H=138.7, M=1.2, Ma=0.16, N=5) : C[5];
EQUATION
    FORALL i IN [1,4](
        CONNECT(C[i].Tqs,  C[i+1].Tqe);
        CONNECT(C[i].Pert,.Pert);
        CONNECT(C[i].Tenc, Tenc);
        CONNECT(C[i].Ray,  Ray);
    )
CONNECT(C[1].Tqe, Tqe);
CONNECT(C[5].Tqs, Tqs);
CONNECT(C[5].Pert, Pert);
CONNECT(C[5].Tenc, Tenc);
CONNECT(C[5].Ray, Ray);
)
```

5.4. COMPOUND MODEL

Here is the complete setup of the water heater. It is reated interactively in ALLAN. The corresponding Modelica text could also be generated.

ALLAN.™

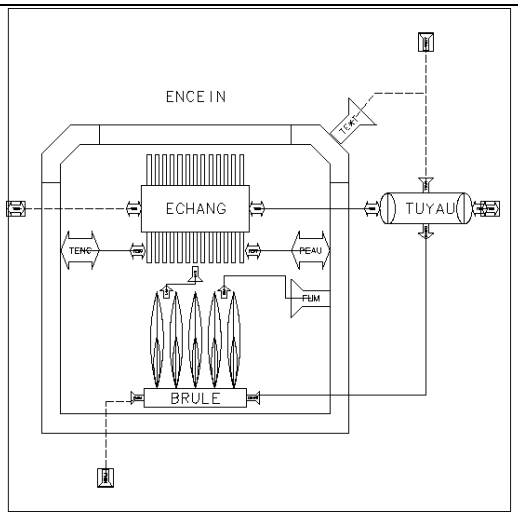


Figure 2 : Internal diagrams of the water heater in ALLAN

Modelica

```
MODEL Chfeau(
    Tq : Tqe, Tqs;
    Entree : Text(start=20, Exact=FALSE);
    Entree : Pnom(start=0, Exact=FALSE),
             Qinfo(start=0, Exact=FALSE);
    Cnddx : Tuyau(F=0.1, H=4, M=0.3, Ma=0.016);
    Cc2d5 : Echang;
    Brule : Brule(Q1=0.06, Q2=0.04, Tb=2);
    Jacket : Encein(Ma=4190);
EQUATION
    CONNECT(Echang.Tenc, Encein.Tenc);
    CONNECT(Echang.Pert, Encein.Peau);
    CONNECT(Echang.Tqs, Tuyau.Tqe);
    CONNECT(Echang.Ray, Brule.Ray);
    CONNECT(Brule.Qinfo, Tuyau.Qinfo);
    CONNECT(Brule.Fum, Encein.Fum);
    CONNECT(Encein.Text, Tuyau.Text);
    CONNECT(Tuyau.Text, Text);
    CONNECT(Echang.Tqe, Tqe);
    CONNECT(Brule.Pnom, Pnom);
)
```

6. CONCLUSION

The Modelica language is still under development. Feedback from potential users is very encouraging. The progress of the design group is presented on the web: URL: http://www.Dynasim.se/modelica/. By now, May 97, Modelica is partially implemented in a parser and it is possible to run some models using a Modelica-to-Dymola translator. Issues such as experience and documentation have only been partially treated and are still on the working table. Complete specifications of the language (continuous, events and documentation) should be available before the end of 1997.

Next year, we very much hope that most of the partners will work on their own tool to implement links to and from Modelica.

Gaz de France's real aim is to encourage modellers to adopt one or more modelling languages which are compatible with each other. As soon as several bodies start using the same language and make an effort to exchange their models, we will not hesitate to use it.

MODELICA DESIGN GROUP

Fabrice Boudaud, Gaz de France, Saint Denis, France / Jan Broenink, University of Twente, The Netherlands / Dag Brück, Dynasim AB, Lund, Sweden / Hilding Elmqvist, Dynasim AB, Lund, Sweden (Dymola) / Thilo Ernst, GMD-FIRST, Berlin, Germany / Peter Fritzon, Linköping University, Sweden / Alexandre Jeandel, Gaz de France, Saint Denis, France (U.L.M) / Kaj Juslin, VTT, Finland / Matthias Klose, Technical University of Berlin, Germany / Francis Lorenz, Lorenz Simulation, Belgium / Sven Erik Mattsson, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden (Omola) / Bernt Nilsson, Department of Automatic Control, Lund Institute of Technology,

Lund, Sweden (Omola) / Martin Otter, Institut für Robotik und Systemdynamik, DLR Oberpfaffenhofen, Germany (Dsblock) / Per Sahlin, Department of Building Sciences, Royal Institute of Technology, Stockholm, Sweden (NMF) / Hubertus Tummescheit, GMD-FIRST, Berlin, Germany / Hans Vangheluwe, Department for Applied mathematics, Biometrics and Process Control, University of Gent, Belgium

REFERENCES

Favret F. 1988 "ALLAN, a general working tool which liberates the user from programming work" In *the Osaka Gas R&D Forum 88* (Osaka, JAPAN, Nov 9-11).

Jeandel A., Larivière E. "Guide de Suivi d'Etude Pour ALLAN.Simulation - version 2" *Gaz de France internal report M.DéGIMA.GSA.888*

Jeandel A., Favret F., Lapenu L., Lariviere E. 1993 "ALLAN.™Simulation, a general tool for model description and simulation." In *Proceedings of the 1993 IBPSA Conference* (Adelaide, AUSTRALIA).

Jeandel A.; Boudaud F. 1997 "ALLAN.™Simulation : description of release 3.1" *Gaz de France report M.DéGIMA.GSA.1887*

Jeandel A.; Ravier P., Buhsing A. 1995 "ULM DOCUMENTATION" (in english or/ou en français) *Gaz de France internal reports M.DéGIMA.GSA 1204 to 1211*

Nakhlé M. 1991 "NEPTUNIX an efficient tool for large size systems simulations" In *2nd International Conference on System Simulation* (Lièges, BELGIUM, Dec 1-3)

Petzold L. R. 1982 "A description of DASSL : A Differential/Algebraic System Solver" SAND82-8637.

Pottier M., 1983 "Extensions et applications envisageables des procédures complémentaires établies pour accéder au progiciel ASTEC 3 : ALLAN 6" Technical report M.D6 n°4034. Gaz de France, DETN, La plaine Saint Denis, FRANCE.

Sahlin P., Sowell E.F. 1989, "A Neutral Model Format for Building Simulation Models" *Proceedings of the 1989 IBPSA Conference* (Vancouver, CANADA).

Sahlin P., Bring A. & Sowell E.F. 1994, "The Neutral Format for Building Simulation" Version 3.01, Swedish Institute of Applied Mathematics, ITM Reports N° 1994:2.

Sahlin P., 1996, "Modelling and Simulation Methods for Modular Continuous Systems in Buildings" Doctoral dissertation, Royal institute of Technology (Stokholm, SWEDEN)