# DESIGNING PROCESS LIBRARIES WITH ECOSIMPRO

Mazaeda Rogelio, Merino Alejandro, Rueda Almudena
Centro de Tecnología Azucarera. Universidad de Valladolid
C/ Real de Burgos. Edificio Alfonso VIII. Planta baja s/n. 47011. Valladolid. Spain
Tel: 983 42 35 63. Fax: 983 42 36 16.
Email: rogelio@cta.uva.es; alejandro@cta.uva.es; almudena@cta.uva.es

## Abstract

*This paper describes how a chemical process library was built and used in the sugar industry and gives a series of key ideas to carry out the work using object oriented languages like EcosimPro. There are detailed illustrations of some parts of the library which describe how certain problems have been tackled, ensuring that the final models are generic, easily reusable and numerically robust.*

**Key words**: Model, Simulation, EcosimPro, library-building.

### 1. INTRODUCTION

The use of dynamic simulation methods in the study of industrial processes is no longer an exception to the rule; it is fast becoming an essential aid in the design and operation of today's industries.

The advantages of using simulation are well known and evident. If a model already exists, it will facilitate decisions concerning the design of the plant to be built. A reliable mathematical model enables us to run simulated experiments which, if carried out on the real process, would prove to be very costly and maybe even dangerous. The safe, economic execution of these experiments would also be invaluable for testing new alternative technologies, and for designing and fine tuning more efficient control strategies. And last, but not least, the use of simulation techniques ensure that technical personnel obtain a better understanding of the complexities of the process which will be very useful when it comes to making the right decisions in the day-to-day operation of the factory.

Despite the foregoing, use on a large scale of dynamic process simulation techniques is still currently limited by the complexity inherent to the development of mathematical models. Nevertheless, recent advances like the emergence of modelling and object oriented simulation tools like EcosimPro make way for a significant improvement in the present situation.

Since object oriented modelling facilitates the reuse of previously built models, it promotes the creation of parametrisable component libraries which are sufficiently general to be used in a multitude of simulation projects. Being able to easily use libraries of well proven, reliable models would evidently give a significant boost to modelling and simulation.

In this paper we present some experiences in designing a library of models which are useful in the sugar industry. The library has been built as the core of a Training Simulator for Control Room Operators in sugar mills [Prada C. et al. 2002], [Merino A., Acebes L.F. 2001], but it can be used for objectives other than training.

The library has a hierarchical structure and includes the physico-chemical properties of many substances, a model of transport elements, a model of elements common to the complete process industry, and models specific to the sugar industry.

The paper is set out as follows: section 2 gives a very brief description of EcosimPro, emphasising aspects of the tool that make it useful in building libraries of reusable models; section 3 explains the general structure of the library; section 4 describes how certain capabilities of EcosimPro have been used in the development of ports; section 5 describes the library of physico-chemical properties; and lastly, section 6 addresses the development of a library of flow elements; for example, the philosophy of work in a matter directly applied to many fields.

### 2. ECOSIMPRO

EcosimPro is designed to simulate dynamic systems using differential and algebraic equations (DAEs), although it can also be used to simulate discrete events.

EcosimPro enables physical systems to be modelled by applying newly developed concepts of object orientation, analogous to those used in programming languages like C++ and Java.

Using EcosimPro's modelling language (EL), the modeller can encapsulate the data and dynamic behaviour of the system in reusable components

which offer a well defined public interface, but which conceal the complexities of the internal workings. The individual components are described by the differential and algebraic equations that define them. More complex components can be created by integrating other, simpler components. It also enables a component to be defined through inheritance properties as an extension or a specialisation of another, more general basic component.

EcosimPro's object oriented modelling technology enables complex dynamic models to be built by interconnecting its constituent components, thus creating a very productive method of work with the reuse of well proven parametrisable components.

Although object orientation in the specific case of dynamic system simulation presents the general distinctive characteristics of this paradigm, logically there are significant differences in respect of conventional object oriented programming. Such differences are determined by the specific nature of the problem to be resolved. In this case it is not a question of generating a code to be run on a particular platform, but of obtaining from the models of individual components a mathematical description correctly presented in the form of a system of differential and algebraic equations for the complete system. A mathematical model built in this way can be studied from different angles, depending on which variables are considered as unknown and which are considered as data. This is a decision to be made by the model user (always with the help of the tool) when defining the so-called partitions. With this information, EcosimPro would be ready to arrange the equations, assigning a computational causality which is deliberately missing in the basic model. To this end, it must be decided what variables are calculated based on what other variables and which equations are used. During this process, blocks of intrinsically related variables may occasionally appear that can only be resolved as a whole; and variables that arise as implicit in non-linear equations or in data tables may also arise which EcosimPro cannot solve. Such variables (and the equations that relate them) form algebraic loops and to resolve them EcosimPro must use the so called tearing algorithms. High index problems may also arise during the process; they appear when there are links between system state variables. Index problems arise in the models for various reasons. They often appear as a result of the links between variables that are generated when different components are connected. They sometimes arise in component models and can be solved by using a different approach to them [EcosimPro manuals and bibliography].

Finally, once the partition has been defined, the model is ready for simulation using the powerful numeric algorithms provided by EcosimPro.

In spite of the advantage of this type of language to automatically generate a mathematical model and the fact that the computational causality is free, the modeller sometimes has to establish certain conditions beforehand or give the model some flexibility so that it adapts to the possible topologies of the object, thus ensuring that the superstructure generated is mathematically robust.

EL is equipped with some reserved sentences and words to help the modeller to direct or restrict the equation arrangement process.

For example, the reserved word EXPL can be used by modifying the declaration of a variable, indicating that it must never be used to break algebraic loops. The sentence INVERSE on the other hand instructs the EcosimPro arrangement algorithm how to correctly invert a certain equation, if such inversion were necessary.

## 3. DEFINING AND ORGANISING A MODEL LIBRARY

The model library developed to date comprises three large parts. The first part defines the group of functions that encapsulate the physical properties required in the models, basically formed by algebraic equations that relate the different variables that may appear in the models.

The second part defines all the possible connectors or types of ports to be used.

The last part contains a group of process unit libraries that constitutes a library of units which are common throughout the whole factory and other units which are specific to the different stages of the production process (diffusion, purification, evaporation, crystallisation, fermentation and boiling).
We are now going to describe these libraries in greater detail — how they have been organised, what has been their philosophy, and the problems encountered with their proposed solutions.

The following explains the philosophy followed in the development of some of these libraries.

## 4. LIBRARY OF PHYSICO-CHEMICAL PROPERTIES

Before running the simulation we must individually insert a series of physical properties of each of the components, as well as of functions, average properties of mixes and global constants.

The following briefly explains how this physico-chemical library has been developed.

- Implementation

- Definition of enumeration types

First of all we have to define all the chemical species which are going to form part of the models to be built.
One feature of EcosimPro that makes it particularly useful for modelling process industry systems is its ability to define products in such a way that each one is formed by a subassembly of all the chemical compounds that it can contain. This allows the model of each elementary unit to be tailor made for the products it receives in each of its ports.
We also have to implement all the subassemblies of chemical species to be used in the different basic operations.

```
ENUM Chemical = {H2O, azucar, marco, impz, sacarosa, cristales, CaO, \
                 O2, N2,CH4, C3H8, CO, CO2, EtOH, PrOH, Iso,\
                 CH3, CH2, OH, CH3OH, CH3CO, CH,\ no_azucar}
SET_OF (Chemical) humos = {O2, N2, CH4, C3H8, CO, CO2, H2O}
```

- Physico-Chemical Properties The next step is to implement the necessary physico-chemical properties of each of the chemical species defined in the enumeration type.

These properties are defined both for the liquid phase and the vapour phase of the chemical compounds.
They are found in the form of equations or tables and are calculated based on the corresponding arguments in each case (pressure, temperature, concentration, etc).

The following is an example of how a property is implemented in equation form:


We have also developed functions in which, with a single call, the average physico-chemical properties of the previously defined different enumeration types can be calculated.

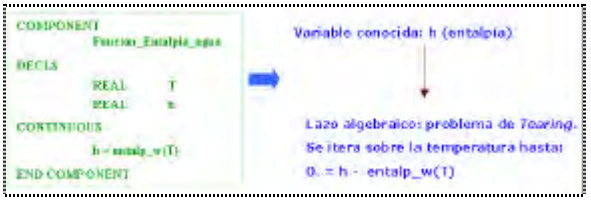
The following example illustrates that this is a single function. This facilitates any subsequent modelling since the parameter SET_OF enables the corresponding property to be calculated for any required enumeration type.


- Problems Encountered and Proposed Solutions

  - Creation of Algebraic Loops

If in a non-linear equation the dependent variable is known rather than the independent variable and the latter cannot be resolved, an algebraic loop is created. In such a case, the corresponding physico-chemical property can be calculated by iterating on the independent variable until the non-linear equation is solved.

For example:

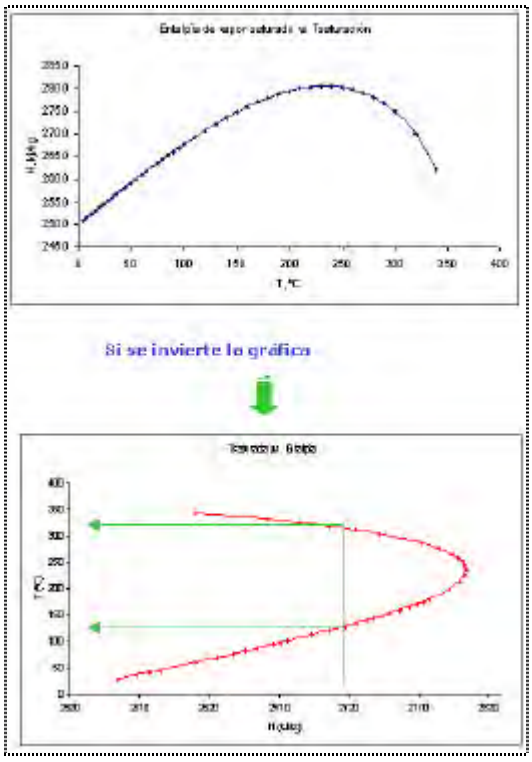



This may give rise to convergence problems at some moment during the simulation and whenever possible must be avoided to facilitate use of the models. The proposed solution to this is to use the INVERSE function which prevents the creation of an algebraic loop:

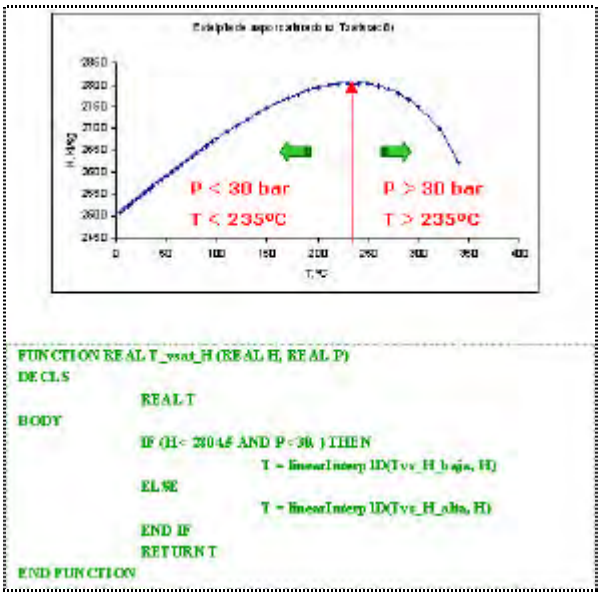
```
COMPONENT          Funcion_Entalpia_agua
DECLS
        REAL       T
        REAL       h
CONTINUOUS
        h = entalp_w(T)
        INVERSE(T) T = T_agua_H(h)
END COMPONENT
```

  - Tables whose inversion results in more than one solution

Another problem to be borne in mind when building a library of models is that a physico-chemical property defined as a table where the dependent variable is known has two possible solutions. These are functions of the following type:




The proposed solution in this case is to invert the table into two segments initially based on two arguments:




```
FUNCTION REAL T_vsat_H (REAL H, REAL P)
DECLS
        REAL T
BODY
        IF (H< 2804.5 AND P<30 ) THEN
                T = linearInterp1D(Tvs_H_baja, H)
        ELSE
                T = linearInterp1D(Tvs_H_alta, H)
        END IF
        RETURN T
END FUNCTION
```

## 5. LIBRARY OF PORTS

Ports are the connection points between the different components. They contain the definition of variables that represent exchange of information between components. Ports avoid having to connect components variable by variable, and having to create separating and connecting components.
In order that the ports can be used as generally as possible, certain EcosimPro characteristics have been used which offer great potentiality and versatility when it comes to reusing components. The liquid or gas ports have therefore been parametrised with SET_OF so that when a connection port is used within a component, we only have to define between parentheses the corresponding enumeration type and the port will have all the properties corresponding to that enumeration type.

The intelligent assignment of properties described above also enables the function calls to be those the user requires based on the parameter SET_OF.

Another EcosimPro feature we have used is the sentence EXPL which prevents certain variables with poor convergence variables from appearing as tearing variables in the algebraic loops, or which prevents EcosimPro from trying to solve them when it comes to arranging the equations. In this way we avoid possible numerical problems.

## 6. MODELLING TRANSPORT COMPONENTS

When building the libraries we saw the need to establish a strategy to develop the components that would appear in the majority of the plant flow lines. This strategy should enable somebody completely uninvolved in building the models to be able to interconnect the library components, and to foresee and prevent the numerical problems associated with these types of connections.

The components that appear in the sugar industry are:

- Pipes
- Valves
- Pumps

Analysis of these components shows that they share a common structure; they comply with the mechanical energy equation. All these components have therefore been modelled so that they comply with the Bernouilli mechanical energy balance.

```
ABSTRACT COMPONENT Bernouilli (SET_OF(Chemical)liquido_mix)
PORTS
  IN        liquido (Mix = liquido_mix)  f_in
  OUT       liquido (Mix = liquido_mix)  f_out
DATA
  REAL      z_in = 0.       "elevacion respecto a la horizontal de la entrada (m)"
  REAL      z_out= 0.       "elevacion respecto a la horizontal de la salida (m)"

DECLS
  REAL hf    "pérdida de carga (m de fluido)"

TOPOLOGY
  PATH f_in TO f_out

CONTINUOUS
  f_out.F = f_in.F              -- condición fluido incompresible
  EXPAND (j IN liquido_mix EXCEPT setofElem(liquido_mix,1)) f_out.C[j] = f_in.C[j]
  (f_in.P *1.e5) + z_in*f_in.Rho*g = \
   (f_out.P*1.e5) + z_out*f_in.Rho*g + hf*f_in.Rho*g -- ecuación Bernouilli (u=cte)

END COMPONENT
```

Based on this abstract component, we developed a hierarchical structure with the different flow elements.
These components have been developed following a series of assumptions:

Incompressible fluid
The fluid temperature remains constant
The fluid compositions remain constant

When these assumptions have been established, the only variables to be related in the models will be the pressure variables or flow variables. Therefore, the relation between the flow and pressure is what will establish the components of the library of flow elements. In general, therefore, the individual components will contribute an associated equation expressed as follows:

$$W = f(\Delta P) \qquad [1]$$

Problems may arise when solving the unknown variables in the equation. Thus, in the simple case of a single flow element in a line, if for example we know the inlet pressure and the flow, and we want to calculate the outlet flow and this variable cannot be solved by the equation [1], the calculation program will leave the equation as implicit and iterate on this value until the equation is closed. This will create an algebraic loop.

As explained previously, algebraic loops can be resolved by EcosimPro, but in our particular case we do not want them to appear because the iteration generated in the search for solutions makes simulation somewhat slower and can sometimes result in convergence problems.

To prevent algebraic loops in situations of this kind, we use the INVERSE sentence structured as follows:

```
x = myFun(y)
INVERSE(y) y = evaluateY(x)
```

What EcosimPro will do is use the first equation and if the unknown variable is that which we use as an argument to INVERT, it will use this other equation. This guarantees that we will not have algebraic loops when we try to solve the equation [1].

### 6.1. CONNECTIONS BETWEEN FLOW ELEMENTS

Mathematical problems also occur when connecting various components of the flow element type, such as high index problems and algebraic loops.

These problems also arise naturally if we try to "manually" calculate some connections between elements of this type.

If, for example, we have a parallel pipe connection and we know the total flow circulating through the two pipes and the line inlet pressure, we should iterate on the flows until the equations for the pressure drop in the two lines are closed. Analysing the system of equations it would be the following:

$$W_1 = f_1(\Delta P_1) \qquad [2]$$
$$W_2 = f_2(\Delta P_2) \qquad [3]$$
$$W_1 + W_2 = W_T \qquad [4]$$

In the first equation the outlet pressure and flow appear as unknown variables. Appearing as unknown variables in the second equation are the outlet pressure which will be equal to that of the first, and the flow which is related to that of the first equation by means of the third equation.

We can see we have a linked system.

$$\begin{array}{l} \\ W_1=f(P_2-P_1) \\ W_2=f(P_2-P_1) \\ W_T = W_1+W_2 \end{array} \quad \begin{array}{c} W_1\ W_2\ P_2 \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

In this case it is not possible to manipulate the equations so that only one equation corresponds to each unknown variable.

To avoid these problems, we have used a method which consists in transforming one of the unknown variables into a state variable so that it passes for calculation. In this way the loop does not appear. We therefore express the equations that relate the flow to the pressure as follows:

$$Alfa^* W1' = f(P2-P1)$$

where alpha is a time constant which must be small so that the equation dynamics are fast and the result is as similar as possible to the algebraic equation. On the other hand, if we make this time constant too

small we are forcing the integrator to use very small steps of integration and therefore alpha shall have to have a compromise value.

As an example of a component that has all these characteristics, we have a pipe.

```
COMPONENT Tubo_liq IS_A Bernouilli (BOOLEAN impl)
DATA
        REAL D                          "diámetro interior del
tubo(m)"
        REAL  L                         "longitud de la
tubería(m)"
        REAL  e = 0.046e-3              "rugosidad de la
tubería. valor por defecto:acero (m)"
        REAL alfa = =0.01               “constante de tiempo
para el cálculo del caudal”
DECLS
        REAL S                          "sección transversal
tubería (m2)"
        REAL f                          "coeficiente de
fricción"

CONTINUOUS
        S = PI/4.* D**2

<eqn1> IMPL (f) f = 1./( -2.5 * log(max(((e/D)*0.27),1.e-8)))**2

EXPAND(impl==TRUE) f_in.W = (S*sqrt(max(0.,hf*(2*g)/(8*f*(L/D))))+0) *
f_in.Rho
         INVERSE (hf) hf = 8.*f*(L/D)*(((f_in.W/f_in.Rho)/S)**2)/2./g

EXPAND(impl==FALSE)   alfa * f_in.W'=
(S*sqrt(max(0.,hf*(2*g)/(8*f*(L/D))))+0)*f_in.Rho - f_in.W
        f_in.T = f_out.T
END COMPONENT
```

We can see how the pipe uses a Boolean parameter which determines the use of one sentence or another using the sentence EXPAND. In this way we can decide whether we want to use the algebraic equation or the differential equation to resolve the problem and thus avoid certain numerical problems.

By building a differential equation into the model we are faced with an added difficulty, this being the index problems.

These occur, for example, when we connect two pipes in series. The same flow circulates through both. Thus, if we use the differential equation for the two pipes we will get the following system of equations:

$$\text{Alfa}* W_1' = f(P_2 - P_1) \qquad [5]$$

$$\text{Alfa}* W_2' = f(P_2 - P_1) \qquad [6]$$

$$W_2 = W_1 \qquad [7]$$

A link is therefore formed between state variables and, as a result, an index problem occurs which the algorithm will solve by creating an algebraic loop.

We are now going to detail the possible types of flow element connections and the numerical problems associated with them.

**Elements Connected in Series**

There can be different cases, depending on what our unknown variables are:

*We know the flow*. In this case the problem is minimal because if we know the inlet flow, the pressure drop in the pipes can be deduced from the type 1 equations.
So if we define the parameter impl = TRUE in the two pipes, the problem is resolved without the need of loops.

The problem is also resolved if we define one of the parameters impl = FALSE, although including a dynamic variable. The problem arises if we connect the two pipes with impl = FALSE. In this case an index problem occurs because, as we mentioned earlier, we are asserting that the two state variables we have are equal.

*We know the inlet pressure*. If the two parameters are TRUE, an algebraic loop is generated because there is no equation to tell it that the pressure drop between the ends is the sum of the pressure drops and that it can calculate the flow on that pressure difference. The program therefore iterates on the flow to obtain the necessary pressures drops in the pipes.

**Elements Connected in Parallel**

As in the case of elements connected in series there can be different cases for elements connected in parallel, depending on which are our unknown variables:

*We know the inlet pressure*. We have no problem in either case because the pressure drops for the two branches will be the same and since we know the pressure at the ends, the flow is calculated for each of the branches.

*We know the flow*. There are two things we can do; if we define the parameter impl = FALSE for both pipes we have a problem because the flow will be linked by equation [4]. If we define them both as TRUE we create a loop with the following calculation reasoning: knowing the flow we do not know the flow that circulates through each of the branches or the inlet pressure; so. we will assume the flow through one of them and with this flow we can calculate the other and with the other flow we can calculate the pressure drop in the two branches; if it is equal we obtain the solution and if it is not we continue to iterate.

Therefore, the way to prevent loops in these types of connections is to define one branch as TRUE and the other as FALSE.

**7. CONCLUSIONS**

As we have seen throughout this presentation, building a library of models using an object oriented modelling language is not limited merely to the

creation of a series of components that are later simply connected. Modellers must bear in mind several aspects; firstly, we must try to make the components as general as possible so that they can be easily re-used; we must also prevent possible problems derived from the use and connection of the models. Flexibility will therefore sometimes have to be limited when it comes to rearranging the equations. Other times, different options must be available through the alternative use of equations of one type or another, building them into the components so that possible mathematical problems in the model can be controlled. Using certain sentences, EcosimPro makes it possible to control our model in this respect. In this paper we have described the creation of a library with these principles; and we have illustrated the specific development of certain libraries such as the physico-chemical library or transport elements library.

**REFERENCES**


Acebes, L.F. (1996). Doctoral Thesis. "SIMPD: Sistema Inteligente de Modelado de Procesos Dinámicos". Departamento de Ingeniería de Sistemas y Automática. Universidad de Valladolid, Spain.

ACSL. http://www.acslsim.com

C. de Prada, F. Acebes, R. Alves, A. Merino, S. Pelayo, A. García, A. Rueda, G. Gutiérrez & M. García. "Un simulador de alcance total para la formación de los operarios de sala de control de factorías azucareras". II Taller Iberoamericano de Informática Industrial. Salamanca. 2002.

EcosimPro. http://www.ecosimpro.com

Merino A., Acebes. L. F. "Modelling and simulation of the diffusion section of a beet sugar process for a plant operator training simulator". ESS`2001 conference. Marseilles 2001.

L.F. Acebes, A. García , A. Merino , S. Pelayo, C. de Prada & A. Rueda "Desarrollo de una librería de modelos de unidades de proceso de la industria azucarera" . Wokshop en Metodología de Modelado y Simulación de Sistemas. Barcelona 2001.

Himmelblau, David M. – Bischoff, Kenneth B. "Análisis y simulación de procesos". ED. REVERTÉ, S.A. BARCELONA. 1976.

Macías Hernández, J. – Feliu Gil, J.A. "La simulación dinámica en ingeniería de procesos". Revista Ingeniería Química. Abril 2000. Pgs.127-131.

Perry & Chilton. "Manual del Ingeniero Químico" Vol.1 y 2. McGraw Hill (1998).

Robert C.Reid, John M.Prausnitz, Bruce E.Poling "The Properties of Gases & Liquids" McGraw-Hill 4ª Edition 1988

Roger G. E. Franks. "Modeling and simulation in chemical engineering." ED. JOHN WILEY & SONS. 1972.

Urquía A. (2000) Doctoral thesis "Modelado orientado a objetos y simulación de sistemas híbridos en el ámbito del control de procesos químicos". Departamento de Informática y Automática Industrial. Facultad de Ciencias. UNED.

William L. Luyben. "Process modeling, simulation and control for chemical engineers". McGraw-Hill. Second Edition. 1990.