INTEGRATION OF AN OBJECT FORMALISM WITHIN A HYBRID DYNAMIC SIMULATION ENVIRONMENT

Jocelyne Perret, Gilles Hétreux, Jean-Marc Le Lann

Laboratoire de Génie Chimique (LGC - UMR 5503), département PSE, groupe Génie Industriel
118 Route de Narbonne, 31077 Toulouse Cedex 04
E-mail: Jocelyne.Perret@ensiacet.fr, Gilles.Hetreux@ensiacet.fr, JeanMarc.LeLann@ensiacet.fr.

Abstract: *PrODHyS* is a general object-oriented environment which provides common and reusable components designed for the development and the management of dynamic simulation of systems engineering. Its major characteristic is its ability to simulate processes described by a hybrid model. In this framework, this paper focuses on the *"Object Differential Petri Net" (ODPN)* formalism integrated within *PrODHyS*. The use of this formalism is illustrated through a didactic example relating to the field of Chemical Process System Engineering (*PSE*).

Keywords: Hybrid Dynamic Systems, Simulation, Petri nets, object-oriented approach, process system engineering.

With the evolution of the computer power, dynamic simulation becomes an essential tool in process design and analysis. It provides an efficient help to process engineers in the knowledge of transient behaviour, for the development of adapted control systems (sensitivity to parameters), or for the monitoring and diagnosis of processes in exploitation. However, the simulation of most processes requires to take into account operating modes often difficult to manage with purely continuous or purely discrete models. In that context, phenomena as abrupt open/closed of valves or material physical state evolution induce discontinuities in the models, involving the notion of *hybrid dynamic systems (HDS)*.
In the *HDS* field, many simulation tools have been developed by researchers. We can mention, for example, *gPROMS* (Barton and Pantelides, 1994), *Shift* (Deshpande *et al.*, 1998), *Omsim* (Andersson, 1994), *Chi* (Fábián *et al.*, 1998), *BaSiP* (Wöllhaf *et al.*, 1996). For our part, we develop since more than ten years a platform named *PrODHyS* (***Pr****ocess* ***O****bject* ***D****ynamic* ***Hy****brid* ***S****imulator*). It is a general object-oriented environment designed for the development and the management of dynamic simulation.

In this framework, this paper deals more specifically with the package relative to the hybrid formalism used to describe devices in *PrODHyS* : the *Object Differential Petri nets (ODPN)*. This formalism combines *Petri nets*, *differential algebraic equations* and *object concepts*. So, the first section makes a general overview of the simulation environment and its software structure. In section 2 and 3, the *ODPN* formalism and the simulation kernel are described. Then, the last sections (5 and 6) illustrate the use of *ODPN* formalism through the example of a typical process introduced in section 4.

1. THE SIMULATION ENVIRONMENT

1.1. Objective of *PrODHyS*

Developed in our research department, *PrODHyS* constitutes the unification of works performed since several years in design and development of object oriented software components dedicated to process simulation (Hétreux *et al*, 2003, Jourda *et al.*, 1996, Moyse, 2000, Perret, 2003, Sargousse, 1999). Although this platform may be connected to a windows graphical user interface (*GUI*) in order to simplify its accessibility to users, *PrODHys* is mostly structured as a library of common building blocks which allow a modular modelling and an equation-oriented simulation of processes. Furthermore, two main features characterise *PrODHyS* :

- As philosophy of this environment is to provide general reusable software components in order to build various kind of complex specialised devices, *PrODHyS* is based on a object-oriented approach which emerges nowadays as an efficient and concrete response to extensibility, reusability and software quality requirements. Each elementary entity is defined

as an abstract object which has to be derived via object mechanisms (inheritance, aggregation, genericity, etc). Then, a *flowsheet* object is defined as a set of hierarchical and recursive *device* objects connected to each other by *port* objects, inside which *material*, *energy* or *information* flows. This "systemic" approach is now clearly established (Hétreux *et al*, 2002, Jourda, 1996, Marquardt, 1992, Nilsson, 1993).

- *PrODHys* associates a hybrid model to each device based on the *ODPN*. This hybrid formalism offers a great level of abstraction in order to reduce the modelling complexity. Each Petri net describes the sequence of continuous states that a device, a recipe or material can reach.

1.2. Software architecture of the platform

The design of *PrODHyS* follows a software development process based on the object technology (*UML, C++*). Currently, this software consists of more than one thousand classes distributed into two layers and seven packages (cf. figure 1) :

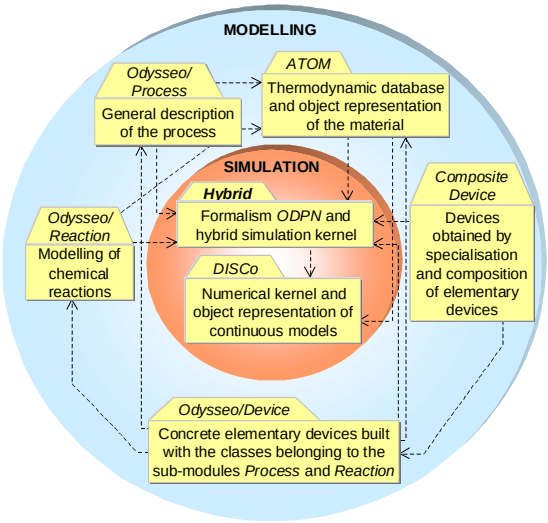

figure 1. Software architecture of PrODHyS

- The internal level corresponds to the *simulation* kernel of the platform. It provides the basic elements allowing the simulation of any dynamic system. This layer includes :
  - the module *Disco* which is the numerical kernel of the system. It allows an object representation of the continuous mathematical models and provides a set of solvers and integrators (*EDA, EANL*).
  - the module *Hybrid* which contains the set of classes used for the description of the *ODPN* formalism as well as the hybrid simulation kernel.
- The higher level includes a set of classes allowing the modelling of processes. The *modelling* layer encapsulates the *simulation* layer and provides a set of general and autonomous entities (classes) which can be exploited by any user who wishes to build its own simulation system or prototype. This level includes :
  - the module *ATOM* which constitutes the thermodynamic data base of the system; it is based on an object representation of the material and allows the computing of thermodynamic properties.
  - the module *Process* which gathers a set of generic and abstract classes, corresponding to a very general description of the process;
  - the module *Reaction* which allows the modelling of chemical reactions;
  - the module *Device* which gathers the "concrete" *elementary* devices.
  - the module *CompositeDevice* which contains devices resulting from the composition and the specialisation of *elementary* devices defined in the module *Devive*.

The main advantage of separating *simulation* level and *modelling* level is to make possible the implementation of platforms dedicated to various fields of applications only by developing the suitable engineering *modelling* layer.

2. HYBRID FORMALISM: THE *ODPN*

2.1. Modelling approaches

Regarding the modelling aspects of processes, two dynamic schemes have to be described : on the one hand, the continuous dynamic, often represented by a set of differential and algebraic equations (*DAE*) and on the other hand, the discrete dynamic, represented by a set of states and transitions. The studies about the combination of continuous and discrete elements have led to several formalisms. A classification is proposed (Zaytoon, 2001) :

- approaches which extend models belonging to the continuous field such as unified model (Branicky *et al.*, 1998) or bond-graphs with switches (Buisson et *al.*, 2001);
- approaches which extend models belonging to the discrete field. We can mention hybrid Petri nets (Le Bail *et al.*, 1991), batch Petri nets (Demongodin, 2001), time Petri nets (Berthomieu et Menasche, 1983), timed automata (Alur et Dill, 1994);
- mixed approaches in which discrete and continuous models collaborate in the same integrated structure. This category concerns hybrid automata (Alur *et al.*, 1995), hybrid statecharts (Kesten and Pnueli, 1992), mixed Petri nets (Valentin-Roubinet, 1999), differential predicate-transition Petri nets (Champagnat *et al.*, 1998).

In our case, a hybrid formalism based on a *mixed approach* and *object concepts* has been adopted : the *Object Differential Petri Nets (ODPN)*.

2.2. Petri Net/Object Oriented paradigm

These last years, many works have emphasized the interest to combine Petri nets (*PN*) and the object-oriented concepts (*OO*). The studies relative to the *PN/OO* paradigm have shown that this association can be performed according to two approaches :

- the first one aims to introduce "*the objects into Petri nets*" (Sibertin-Blanc, 1985). The subjacent philosophy is to model a sub-system by a single Petri net which handles individualised tokens carrying information. In this approach, each token is a generic entity defined by an object class made up of both a set of attributes (including state variables) and a set of methods which deals with these data (including equations). Consequently, the Petri net models the control structure of the system (its general behaviour) whereas the tokens represent the associated data structure (a particular version of this system). This mechanism, with individualisation of the tokens, makes the network more compact without information loss.
- the second approach is based on "*the introduction of Petri nets into objects*" (Paludetto, 1990). This approach enables to describe the internal behaviour of the object. The marking of the Petri net indicates the current state of the object, the firing of a transition involves the execution of one of its methods and the global structure of the net specifies the legal execution sequences of the methods.

In fact, these two approaches are not incompatible but complementary (Bastide, 1995). For this reason, the "*extended combined approach*" has been defined as an extension of the previous ones (cf. figure 2).

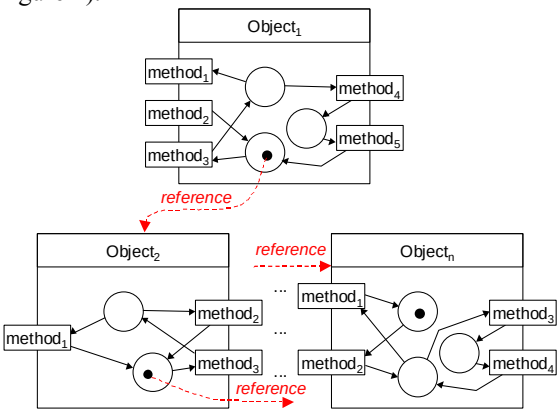

figure 2. Extended combined approach

It consists in introducing an infinite number of encapsulation levels of objects and Petri nets, thus allowing a complete integration of the *PN/OO* paradigm. Moreover, this approach makes possible the definition of hierarchical points of view of the system.

2.3. The dynamic and behavioural aspects

The modelling of the dynamic part of a system is based on the *Differential Predicate-Transition Petri nets* concepts. The *ODPN* model is characterised by two kind of places :

- the *discrete* places : it represents a control, an operation, the availability of a resource or quite simply, a discrete state,
- the *differential* places : it indicates a continuous behaviour whose dynamic is governed by a differential algebraic equations system. Thus, the marking of a *differential* place starts up the continuous evolution of state variables.

2.4. The static and structural aspects

To describe the static part of the system, the *ODPN* model is based on the object-oriented (*OO*) concepts. Indeed, encapsulation, inheritance, composition and polymorphism concepts lead to the definition of entities which are at the same time strongly consistent and slightly coupled with their environment, which increases their reuse possibilities. In addition, the structuring with classes generates elements whose access is perfectly controlled and specified through an interface.
Furthermore, the inheritance and composition mechanism are extended to the Petri net included in the class definition. When a new class object is created by aggregation of others classes, a new Petri net is generally created in the composite object in order to control the Petri net of the components (inducing a master/slave relationship). The concept of *transition merging* (Champagnat *et al.*, 1998) is also used when it is about an association relationship. Finally, in the case of inheritance relationship, the "daughter" class inherits the net of the "mother" and if necessary, this net may be specialized by adding new places and/or transitions.

2.5. Semantic of the *ODPN* formalism

Each entity of the system is described by an *object class* made up of attributes (including the continuous state variables) and methods.

In order to describe the dynamic and structural aspects of the systems, the *object differential Petri nets* is defined by a set of constitutive elements illustrated on figure 3. It includes :

- *Places :*

The *ODPN* model is characterised by a set of *discrete* places (single circle) and *differential* places (depicted with two concentric octagons).

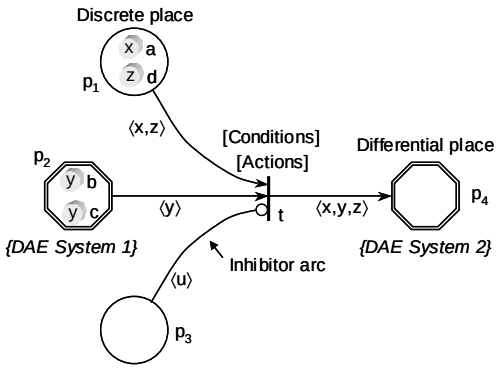

figure 3. Semantic of the model

- *Tokens*

Each token is defined as an instance of class. So, it encapsulates the entity characteristics and its own state variables. For this reason, it does not exist any global variable. Because of the extended combined approach, the internal behaviour of each token may be described by a Petri net. Furthermore, tokens move on a Petri net according to formal variables carried by the arcs.

- *Formal variables*

A formal variable is typed by an object class that specifies the type of tokens authorised to replace it. Let us note c_i, an object class; a formal variable typed by the class c_i is noted $\langle c_i \rangle$; n_k formal variables typed by the same class c_i are noted $\langle c_i^{(k)} \rangle$ (for $k = 1, n_k$). Moreover, a tuple of formal variables is noted $\langle c_1, c_2, \ldots, c_n \rangle$.

In order to illustrate this notion, let us consider the example shown on figure 4.

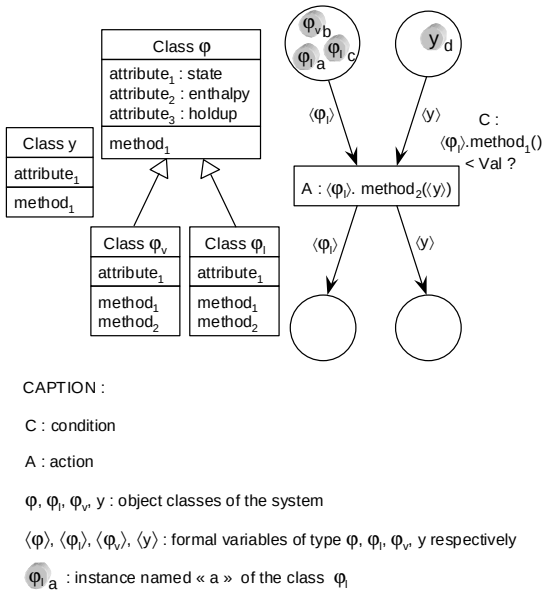

figure 4. Formal variables

Three formal variables are defined $\langle \varphi_l \rangle$, $\langle \varphi_v \rangle$ and $\langle y \rangle$; they are typed by the classes φ_l, φ_v and y respectively. They can be replaced only by tokens carrying the same inscription. As a result, $\langle \varphi_l \rangle$ can be replaced by the tokens a and c, $\langle \varphi_v \rangle$ can be replaced by the token b and $\langle y \rangle$ can be replaced by the token d. Let us notice that for a formal variable $\langle \varphi \rangle$ of type φ (where φ is the mother class of classes φ_l et φ_v), the substitution by tokens a, b and c is feasible thanks to the inheritance and polymorphism principles.

- *Arcs*

The arc constitutes the link between the place and the transition. In order to ensure the consistency of the model, input and output arcs are explicitly typed (figure 4). Indeed, they carry information allowing to specify the token classes authorised to forward through the arc. The inscriptions carried by the arcs correspond to formal variables; thus, any token authorised to move on an arc replaces the formal variable of the same type.
Moreover, an object differential Petri net also holds inhibitor arcs. An inhibitor arc is an arc that allows to test the lack of tokens on the place located above.

- *Transitions*

Each transition is characterised by a set of *conditions* (also called *enabling functions*) and *actions* (also called *junction functions*).
Conditions are made up of attributes and methods carried by the tokens enabling the transition and/or coming from the object which is the owner of this Petri net (i.e. the object whose behaviour is described by this Petri net).
In the same way, *actions* perform the methods offered by the tokens crossing the transition and/or methods belonging to the object which is the owner of this Petri net. Let us note that the aim of these actions not only consists in computing the initial values of the continuous variables, but also in modifying the state of tokens.

3. THE SIMULATION KERNEL

In order to implement the *ODPN* formalism, the simulation kernel is break down into three modules: the *discrete solver* (Petri net token player), the *continuous solver* (integrator based on the *Gear* method (Gear, 1971)) and a *simulation manager* which manages the interactions between the two solvers. The operating cycle of the simulator is then the following:

1. First, the simulation manager builds the global continuous model corresponding to the initial marking of each Petri net and initialises all state variables ;
2. The discrete solver plays the Petri nets until no more transition can be fired. During this step, the actions of each fired transitions are performed (among others, the initial values of states variables and their derivatives are computed).

3. Then, the simulation manager concatenates the *DAE* systems associated with each marked differential place as well as the conditions of each enabled transition;
4. Just before the integration, the continuous solver gets the ability to automatically calculate new consistent initial values for state variables, if necessary. Then, it performs the integration of the resulting global *DAE* system, involving the continuous evolution of the state variables. At the same time, the conditions of enabled transitions are monitored. The continuous solver is stopped as soon as an event occurs, i.e. an enabling function becomes true.
5. Here, the control is given to the discrete solver; the transition associated with this event is fired and its actions are performed. Then, it sets up a new marking and return to **2**.

4. ILLUSTRATIVE EXAMPLE CONSIDERED

A didactic example, shown on figure 5, has been chosen in order to illustrate the concepts implemented within *PrODHyS*.

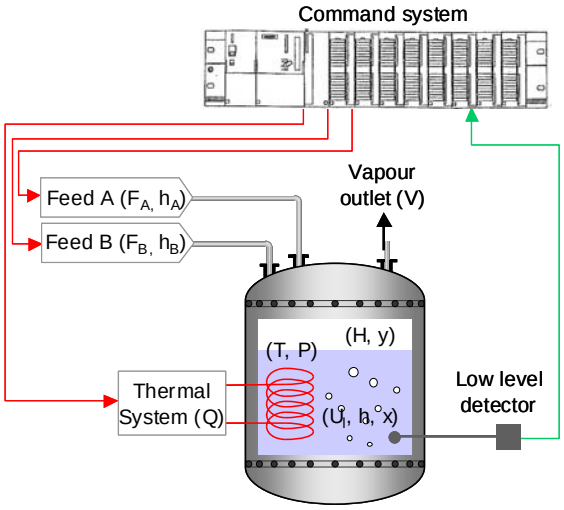

figure 5. Didactic example

It consists of a tank equiped with a low-level detector and a thermal system allowing the heating of the liquid phase (with an energy Q). This tank may be filled via two material feeds A and B characterized respectively by the data (F_A, h_A, x_A) and (F_B, h_B, x_B), where F_f, h_f and x_f are respectively the flow, the liquid enthalpy and the liquid composition vector of feed f (f=A,B). According to the operating conditions (i.e. boiling point reached), the liquid phase may flash with a vapour flow V.
Variable U_l represents the liquid holdup in the tank. The outlet vapour is open on the outside. So, the pressure P is supposed to be constant and the vapour holdup U_v is neglected in front of U_l. Moreover, if a vapour phase exists, it is supposed it disappears as soon as the heating is stopped (which is close to the physical meaning, considering the ultra-fast dynamic of this hydraulic phenomenon).

The mathematical model of this system at the thermodynamic equilibrium and in its maximal state (i.e., liquid/vapour) is as follows :

$$\frac{dU_l}{dt} - F_A - F_B + V = 0 \quad (1)$$

$$\frac{d(U_l.x_i)}{dt} - F_A x_{A_i} - F_B x_{B_i} + V y_i = 0 \quad (i = 1, n_c) \quad (2)$$

$$\frac{d(U_l h)}{dt} - F_A h_A - F_B h_B + VH - Q = 0 \quad (3)$$

$$L - \frac{U_l V_{ml}}{S_c} = 0 \quad (4)$$

$$y_i - K_i.x_i = 0 \quad (i = 1, n_c) \quad (5)$$

$$\sum_{i=1}^{n_c} (x_i - y_i) = 0 \quad (6)$$

$$K_i - mK_i(T, P, x, y) = 0 \quad (i = 1, n_c) \quad (7)$$

$$h - mh(T, P, x) = 0 \quad (8)$$

$$H - mH(T, P, y) = 0 \quad (9)$$

$$V_{ml} - mV_{ml}(T, P, x) = 0 \quad (10)$$

$$V_{mv} - mV_{mv}(T, P, y) = 0 \quad (11)$$

Equations (1), (2) and (3) represent respectively the global material balance, the partial material balances (n_c : number of pure component) and energy balance. Variable h is the liquid phase enthalpy and variable H, the vapour phase enthalpy. Equation (4) determines the liquid level L according to U_l, the tank area S_c and the molar volume of the liquid phase V_{ml}. Equations (5) and (6) represent the liquid/vapour equilibrium, where x and y are the composition vectors of respectively, the liquid phase and the vapour phase (x_i or y_i is the i^{th} element of x or y). Finally, equations (7), (8), (9), (10) and (11) are the models (denoted mC(...) for the constant *C*) used for the liquid/vapour equilibrium constants K_i, the liquid enthalpy h, the vapour enthalpy H, the liquid molar volume V_{ml} and the vapour molar volume V_{mv} within the tank.

This batch process follows the basic recipe described by the GRAFCET of figure 6 :

1. first, the tank is filled via the input feeds A and B for the durations d_A and d_B respectively,
2. then, a heating operation is started and maintained until the liquid level L reaches the threshold detected by the sensor (LOW_LEVEL). Indeed, as soon as the boiling point is reached, a vapour flow V appears and gradually reduces the liquid holdup U_l.

It is important to note that in such a system, two kind of event will induce a model commutation :

- commutations known as *controlled* resulting from a signal emitted by the control device and

appearing explicitly on the GRAFCET (FEED_A, FEED_B, HEAT),

- commutations known as *autonomous* resulting from the intrinsic evolution of the system (here, the material : monophasic to diphasic state). In this case, they do not have to appear on the GRAFCET.

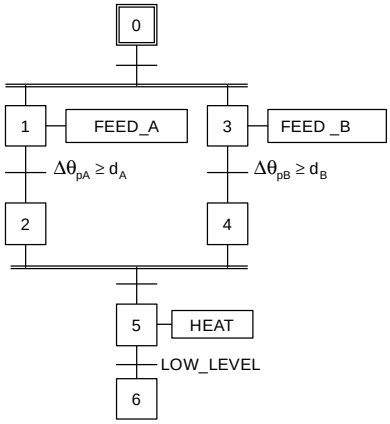

figure 6. Recipe associated with the example

5. OBJECT MODELLING OF A PROCESS

5.1. Structure of the simulation model

To carry out the simulation of such a system, it is necessary to model the *command* part (regulator or automaton), the *operative* part (the process), and the *communication* part (exchanged signals). As a result, a hierarchical structure of models is defined (figure 7).

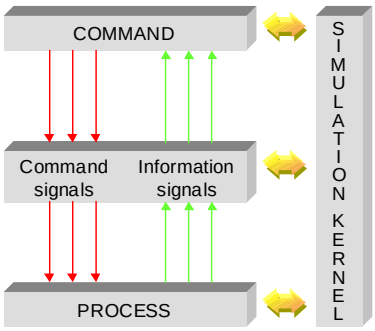

figure 7. Structure of the simulation model

The *master* model associated to the command level represents the recipe. The *slave* models of the process level describe the behaviour of the process in reaction to orders of the higher level. The communication part is characterised by signals exchanges which ensure the link between these two levels. It is based on the use of both information signals coming from the process and command signals coming from the higher level. The representation of these three parts is embedded in the *ODPN* model.

5.2. Topology of the flowsheet

In a general way, a device is defined as an enclosure which can exchange material, energy or information. In order to formalise these exchanges, an interface element named *port* has been introduced. According to the nature of the exchanges, two kinds of ports are identified:

- *communication* ports which allow an information exchange,
- *transport* ports which allow a physical exchange of either material or energy. In this case, the transfer is characterised by a *flow* and a *potential*.

Defining the *topology* of a device consists in determining the type and the number of ports that the device owns. Figure 8 gives the adopted model in order to represent the topology of the illustrative example.

Here, the tank owns two material input ports (connected to the feedings A and B respectively), an energy input port (connected to the heating system), a communication output port (to reach the liquid level in the tank) and a material output port (for the vapour outlet) connected to the surroundings.

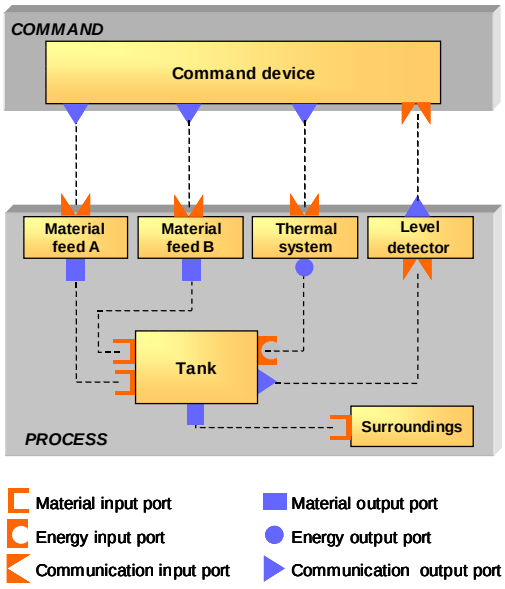

figure 8. Topology of the flowsheet

5.3. Modelling of the exchanged signals

The modelling of the exchanged signals within *PrODHyS* is achieved in an explicit way. Indeed, the device variables are encapsulated within the objects and are made unreachable from the outside. In a general way, the exchanged signals result from either the emission of a command or the reception of an information. In both cases, the nature of the exchanged signal is binary, discrete or real (cf. figure 9).

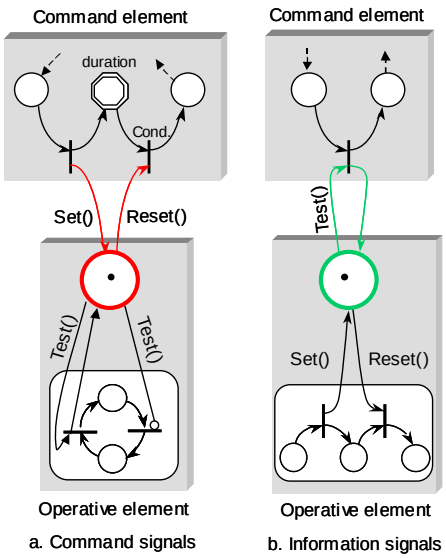

figure 9. Exchanged signals

The signal (input or output) is thus modelled by a discrete place called *signal place* which only owns binary or information tokens; the state of the signal is then associated with the marking of this place. These places allow the distinction between the *active* devices (controlled) such as the material feed and the passive devices (not controlled) such as the tank (see figure 8).
The *command signal* is usually managed by using two kind of arcs : *Set()* and *Reset()*; they are linked to the command element (the recipe Petri net for instance). Thus, while this place is marked, the corresponding command is supposed to be set. The operative elements get this signal thanks to the arcs *Test()*. Regarding the *information signal*, the marking of the discrete place informs the command level about the occurrence of an event on the process level.

5.4. Modelling of the command level

The command level describes the recipe which the process has to follow. On this level, the continuous aspects are often reduced to the explicit expression of operation durations.

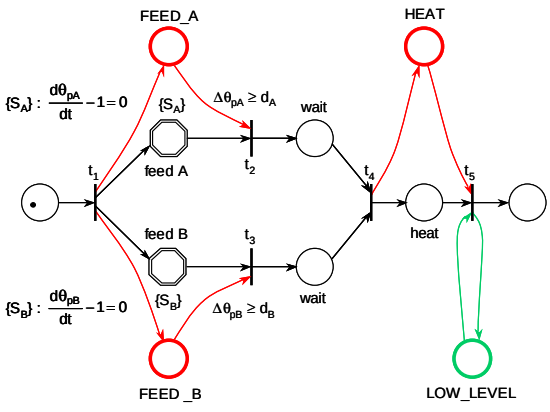

figure 10. Recipe with exchanged signals

The recipe of figure 6 (GRAFCET) can now be modelled by the Petri net of figure 10. It represents the command signals enabling to start the feeding and the heating of the tank and the information signals informing the command level about the crossing of the low level.

5.5. Modelling of the process level

The problematic of the hybrid modelling lies primarily in the management of the legal sequences between the various possible configurations of the model, i.e. the resulting differential algebraic system. When a process becomes complex, the number of possible states and thus, the number of configurations to be managed, may quickly be significant, leading to a combinatory explosion. The system of the illustrative example is made up of twelve configurations. Each one owns a specific differential algebraic equations system. These combinations are related to the "open" or "closed" states, the feedings (none, A, B, A and B), the heating of the tank and the monophasic or diphasic state of the material. The resulting Petri net is shown on figure 11.

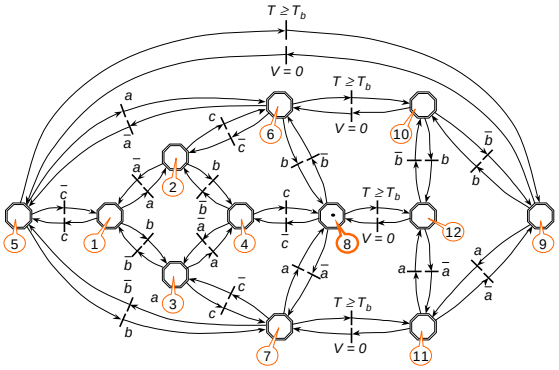

figure 11. Petri net associated with the example

Each configuration appears explicitly as a place. The events associated with the transitions are also indicated: a and b correspond to the feedings A and B; c corresponds to the heating state. The internal temperature is also monitored. The reach of the bubble point (namely represented by the following condition $T = T_b$) leads to a physical state change of the material and involves the crossing of a transition. In the same way, the vapour flow V is monitored and induces a physical state change of the material when it becomes equal to zero (liquid/vapour to liquid). On figure 11, the current state indicates that the feedings A and B are open, the heating is switched on and the material state is liquid.
In fact, the resulting Petri net offers a monolithic and not structured representation. This makes it not easily exploitable even on this simple but tricky example and limits the potential reuse of the already developed models.

6. DECOMPOSITION OF THE MODEL

The integration of the object philosophy in *PrODHyS* allows to get round this difficulty. Indeed, several authors have shown that it was possible to build most of *complex* devices by composing and/or specialising a set of *elementary* devices. All the difficulty lies in the characterisation of these elementary devices which are not necessarily real devices, but rather abstract, autonomous, and/or generic entities, having a simple and predefined functionality and communicating via an isomorphous interface. The following sections specify the modelling concepts of the process level and show the way the extended combined approach is used.

6.1. Modelling of the material

6.1.1. The object "material"

The philosophy adopted in *PrODHyS* is to slightly couple material with the device which contains it (based on an association relationship). Rather than joining the material behaviour with the device one, each element is separately described. The advantage of this decomposition is to create a material object reusable in any other system :

- First, the global continuous model associated with each configuration of the system is split into two subsets:
  - equations (1) to (4), which depend on the tank and on its inlets and outlets, are assigned to the tank;
  - the others are assigned to the material.
- In the same way, the set of variables is split into two subsets:
  - variables F_A, F_B, V, Q which are the inlets and outlets of the tank and L which depends on the tank geometry, are associated with the tank;
  - the others (P, T, x_i, y_i, H, h, V_{ml}, V_{mv}, K_i) are assigned to the material.

Thus, in order to dissociate the tank and the material behaviour, each subsystem owns a specific Petri net (cf. figure 12). However, the material behaviour remains integrated into the tank model thanks to a typed token m carrying the material object and moving on the tank Petri net.
So, the global continuous model of the set "tank/material" results from the concatenation of the differential algebraic system attached to the current state of the material Petri net with the differential algebraic system attached to the current state of the device Petri net.
Figure 12 represents, at the top, the object Tank whose behaviour is specified by the Petri net TankPN. This one consists of eight places, each one modelling one of the continuous states of the tank. The Binary token of the Petri net of figure 11 is now

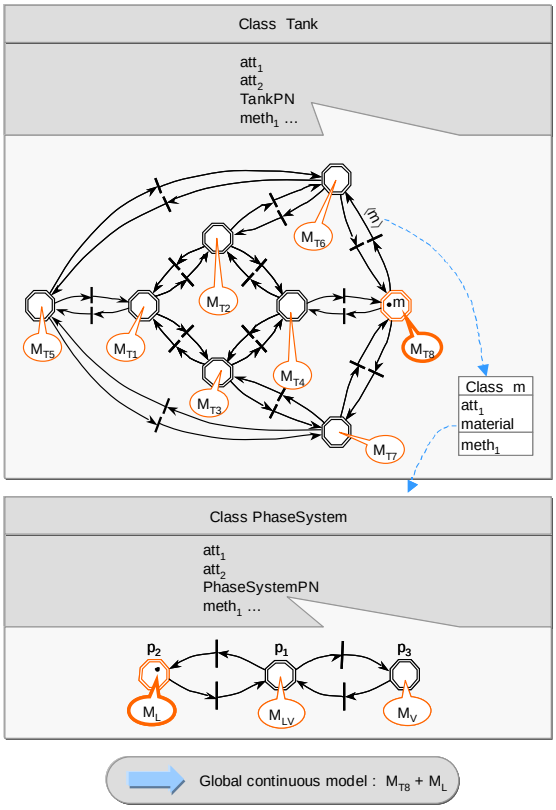

figure 12. Tank / Phases *system*

replaced by a Material token. The bottom of figure 12 represents the Material object whose behaviour is described by the Petri net PhaseSystemPN. For this example, it is only made up of three places corresponding to the liquid state, the vapour state and the liquid/vapour state respectively. In order to avoid overloading the Petri net, let us note that the material type appears only on one arc of the tank Petri net, but it should be represented in fact on all the arcs of the Petri net.
The current configuration of the global system shown on figure 11 (i.e. feedings A and B, heating and liquid state) corresponds to the marking indicated on figure 12. Thus, the continuous model results from the concatenation of the *DAE* system M_{T8} explicitly associated with the marked place p_8 of the tank Petri net and the *DAE* system M_L linked to the Material token marking this place; indeed, the model M_L is associated with the marked place p_2 of the phases system Petri net (cf. figure 12).

6.1.2. The object "phase"

In fact, material is made up of a set of phases, each one being characterised by specific variables. Indeed, the phase is usually described by qualitative physical properties such as enthalpy, molar volume, viscosity, etc. and by quantitative ones such as molar or volumic holdup. It owns, consequently, the equations allowing to determine these variables.
The phases system is supposed to be homogeneous and at the thermodynamic equilibrium. So, it is

characterised by specific variables, the temperature and the pressure, and by the equations relative to the equilibrium between phases.

In order to improve clearness and modularity, the model relative to the material is broken up:

- a part gathers the equations and variables belonging to the set "phases/phases system". The corresponding model is then associated with the differential places of the phases system Petri net. For example, the place corresponding to the liquid/vapour state contains the equations representing the liquid/vapour equilibrium; on the other hand, the place corresponding to the monophasic state does not own any particular equation.
- the other part of the model gathers the equations and variables allowing the computation of the phases physical properties; they are associated with the differential places of the phases Petri nets.

Thus, a new kind of token has been created: it is called Phase token and carries a Phase object. Among its attributes, this object defines a Petri net which describes its behaviour. The equations relative to the phase are thus associated with the places of the Phase Petri net. However, the phase model remains integrated into the material model thanks to the Phase token which moves on the PhaseSystem Petri net. According to the kind of the carried phase, these tokens are characterised by a specific type which specifies if it is about a vapour phase φ_v or a liquid phase φ_l.

So, the class PhaseSystem described on figure 12 is replaced by the model of figure 13. The Tank object is represented at the top and its behaviour is described by the attribute TankPN. It is still made up of eight places, each one describing one of the continuous states previously defined. The Material token moves on the tank Petri net and its behaviour is described by the Petri net PhaseSystemPN. Now, it owns only two places: the first one symbolises the monophasic state of the material whose nature depends on the token marking the place; the second one symbolises the liquid/vapour state indicating the presence of the liquid and vapour phases within the phases system. Regarding the Phase tokens, they move on the Petri net PhaseSystemPN and own a specific Petri net called PhasePN. This one is made up of two places corresponding to two distinct configurations (not detailed here).
Considering the same current state (feedings A and B, heating and liquid state), the marking of the Petri nets is indicated on figure 13. The continuous model results from the concatenation of the *DAE* system M_{T8} associated with the marked place p_8 of the tank Petri net, the *DAE* system M_M associated

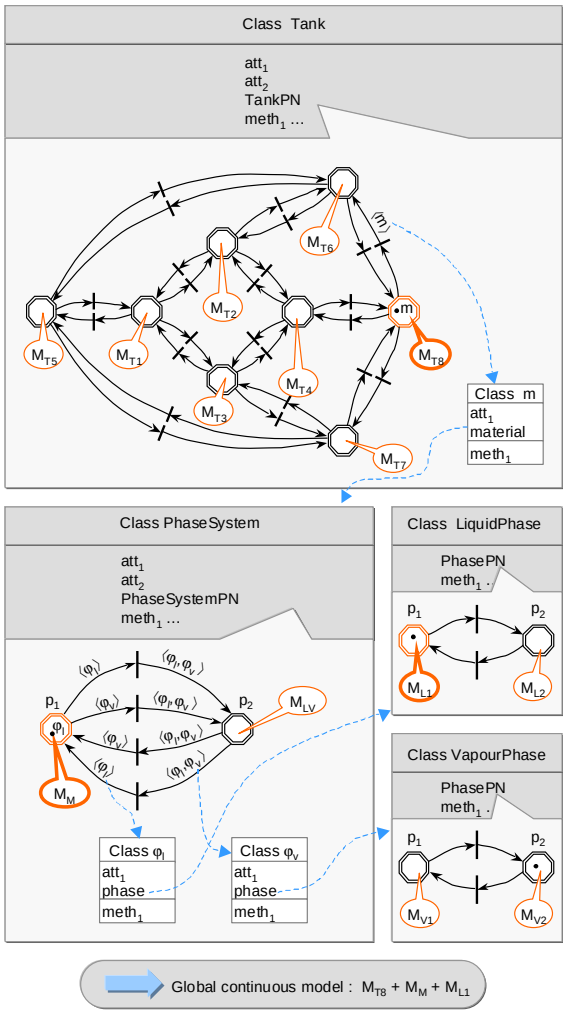

figure 13. Tank / Phases system / Phases

with the marked place p_1 of the phases system Petri net and finally the *DAE* system M_{L1} associated with the marked place p_1 of the liquid phase Petri net. Indeed, as material is in a liquid state, only the Phase token carrying the liquid phase marks the monophasic differential place of the phases system; the Phase token carrying the vapour phase does not exist.

6.2. Modelling of ports

The number and the type of ports may quickly increase the combinatory and so the complexity of the model. Moreover, this thwarts our objective which is to create generic and reusable models.
For these reasons, another type of token has been created. It is called Port token and carries a Port object. The behaviour of the port is then described by a specific Petri net named PortPN and defined as an attribute of the Port object. This one is characterised by two places : a discrete place which indicates an inactive state and a differential place which indicates an active state.

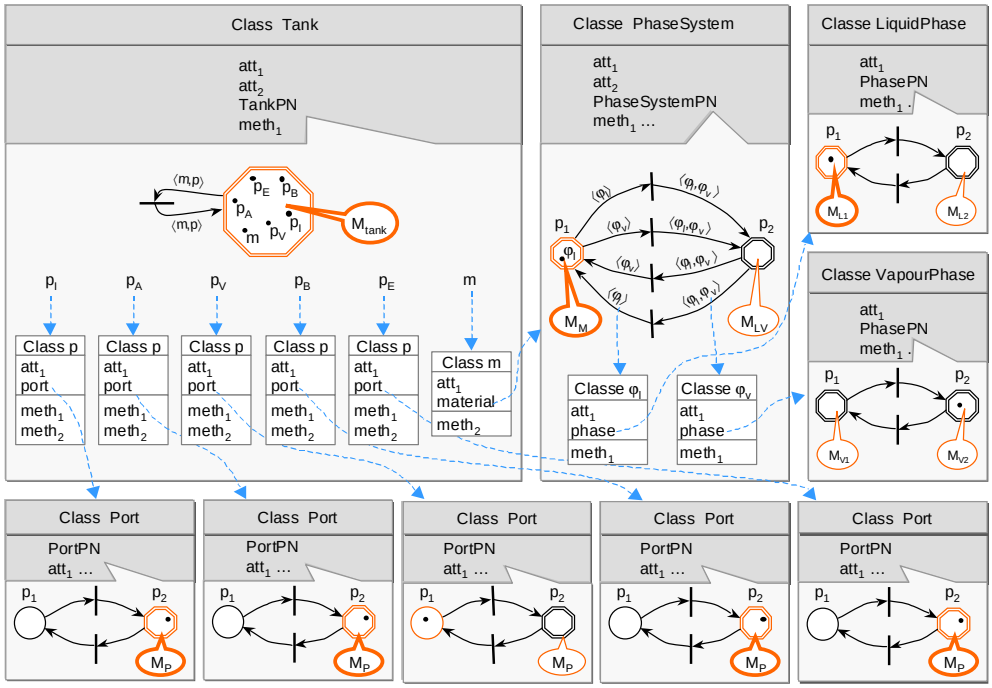

figure14. Tank/Phases system/Phases/Ports

Thus, in our example, the flow variable is no longer directly associated with the tank but is now carried by the port. The combinatory of the model, related to the inlets and outlets, is thus reduced by replacing the various induced states by only one state, marked by a set of Port tokens.
Regarding transport ports, the *active* state supposes that the associated flow belongs to the set of unknown variables of the system. On the contrary, when the state is *inactive*, flow becomes a parameter equal to zero. The balance equations associated with the continuous model of the tank Petri net remain the same whatever the active or inactive state of its input and output ports.

The model associated with the system of the illustrative example is then made up of a set of Petri nets represented in figure 14. The tank is now described by only one place. The transition allows to detect a possible event. The tokens associated with the ports of the device mark the differential place. They are five: two input material Port tokens p_A and p_B, one output material Port token for the vapour outlet p_V, one energy Port token p_E and finally, one communication Port token p_I. Each one owns a specific Petri net.

Considering the same current state, the marking of the set of Petri nets is indicated in figure 14. The current places of Petri nets associated with the input material ports and the energy port are the differential ones. The associated flows are thus taken into account in the material and energy balances. On the contrary, the current place of the Petri net associated with the output material port is the discrete one: the associated flow appears in the balances but as a zero-parameter. The global continuous model associated with the current configuration of the system is then the following one:

$$\frac{dU_l}{dt} - F_A - F_B + \cancel{W} = 0$$
$$\frac{d(U_l.x_i)}{dt} - F_A x_{Ai} - F_B x_{Bi} + \cancel{W} y_i = 0$$
$$\frac{d(U_l.h)}{dt} - F_A h_A - F_B h_B + \cancel{W} H - Q = 0$$
$$L - \frac{U_l V_{ml}}{S_c} = 0$$
$$h - mh(T,P,x) = 0$$
$$V_{ml} - mV_{ml}(T,P,x) = 0$$

M_{Tank} (first four equations) ; M_{L1} (last two equations)

The model associated with the monophasic place of the phases system Petri net is empty; it does not contain any particular equation. In this example, the models of the ports are also empty. However, they indicate if the flow variable is considered as an unknown variable or as a parameter and nullify it if this last case is checked. On the other hand, the eight configurations related to the inputs/outputs of the tank are now replaced by only one state and the associated models are replaced by a single model M_{Tank} for which all flows are represented. The above equations system highlights the flows defined as unknown variables (surrounded) and those defined as zero-parameters (barred with a cross).

In addition, let us emphasise that the model M_{Tank} is generic; it is set up according to a general process which depends on the marking of the differential place. Indeed, it obtains the variables and parameters that it needs via the tokens marking the

place. The model is thus formulated by using the formal variables $\langle m \rangle$, $\langle \varphi \rangle$, $\langle p \rangle$, which are replaced by the Material, Phase and Port tokens respectively when the place is marked. In order to illustrate these remarks, the following differential algebraic system represents how the continuous model M_{Tank} associated with the tank is defined.

$$\frac{d\left(\sum_{k=1}^{n_\varphi} \langle m \rangle . \langle \varphi^{(k)} \rangle . U\right)}{dt} - \sum_{k=1}^{n_{pme}} \langle p^{(k)} \rangle . F + \sum_{k=1}^{n_{pms}} \langle p^{(k)} \rangle . F = 0 \qquad (12)$$

$$\frac{d\left(\sum_{k=1}^{n_\varphi} (\langle m \rangle . \langle \varphi^{(k)} \rangle . U)(\langle m \rangle . \langle \varphi^{(k)} \rangle . z_i)\right)}{dt} - \sum_{k=1}^{n_{pme}} (\langle p^{(k)} \rangle . F)(\langle p^{(k)} \rangle . z_i) + \sum_{k=1}^{n_{pms}} (\langle p^{(k)} \rangle . F)(\langle p^{(k)} \rangle . z_i) = 0 \quad i = 1, n_c \qquad (13)$$

$$\frac{d\left(\sum_{k=1}^{n_\varphi} (\langle m \rangle . \langle \varphi^{(k)} \rangle . U)(\langle m \rangle . \langle \varphi^{(k)} \rangle . h)\right)}{dt} - \sum_{k=1}^{n_{pme}} (\langle p^{(k)} \rangle . F)(\langle p^{(k)} \rangle . h) - \sum_{k=1}^{n_{pee}} \langle p^{(k)} \rangle . Q + \sum_{k=1}^{n_{pms}} (\langle p^{(k)} \rangle . F)(\langle p^{(k)} \rangle . h) + \sum_{k=1}^{n_{pes}} \langle p^{(k)} \rangle . Q = 0 \qquad (14)$$

$$L - \frac{(\langle m \rangle . \langle \varphi_l \rangle . U)(\langle m \rangle . \langle \varphi_l \rangle . V_m)}{S_c} = 0 \qquad (15)$$

$$h - mh(T, P, x) = 0 \qquad (16)$$

$$V_m - mV_m(T, P, x) = 0 \qquad (17)$$

where:

n_{pme}/n_{pms} : number of input and output material ports;
n_{pee}/n_{pes} : number of input and output energy ports ;
n_φ : number of phases of the Material token;
U : phase holdup;
F : material flow;
z_i : molar fraction of the component i;
Q : energy flow ;
h : phase enthalpy ;
L : liquid height ;
V_m : molar volume.

Equations (12), (13) and (14) are the material and energy balances respectively. Equations (16) and (17) are the models used for the molar enthalpy and the molar volume. Finally, equation (15) enables to compute the liquid height in respect with the tank geometry. Here, the liquid level is supposed to be calculated in the cylindrical part of the tank. If this calculation has to take into account the real shape of the tank, several distinct equations would be necessary. In this case, this would induce that the tank Petri net would be composed of several places.

CONCLUSION

The modelling of most industrial processes requires to take into account hybrid phenomena especially in chemical Process Systems Engineering. The use of a high level model associated with powerful numerical integration methods allows to build a robust hybrid dynamic simulator. Designed according to an object approach, *PrODHyS* provides a library of autonomous software components that may be gathered or specialised in order to develop a specific device. For a developer, the exploitation of these elementary components allows to speed up the design and the implementation of a new device. For a user, the exploitation of the predefined devices offered as *"black boxes"* makes easier the setting up of simulation campaigns. The continuous part of *PrODHyS* has been used in the *OPERA* project for real time operators training simulation (OPERA, 1999). Concerning the new hybrid aspect, *PrODHyS* has been used with success for the simulation of several large systems such as a reactive distillation column (model made up more than 600 equations, (Perret *et al*, 2003)). These elements make *PrODHyS* an operational and evolutive tool.

REFERENCES

Alur R., Dill D.L. (1994). *A Theory of Timed Automata,* Theoretical Computer Science, Vol.126, N°2, p.183-225.

Alur R., Courcoubetis C., Halbwachs N., Henzinger T.A., Ho P.H., Nicollin X., A. Olivero, Sifakis J. and Yovine S. (1995). *The algorithmic analysis of hybrid systems*, Theoretical Computer Science, Vol.138, p.3-34.

Andersson M. (1994). *Object-Oriented Modelling and Simulation of Hybrid Systems*, Ph.D thesis, Lund Institute of Technology, Sweden.

Barton P.I., Pantelides C.C. (1994). *The Modelling of Combined Discrete/Continuous Processes*, AIChE Journal, 40:966-979.

Bastide R. (1995). *Approaches in unifying Petri nets and the Object-Oriented Approach*, Application and Theory of Petri Nets - Workshop on Object-Oriented Programming and Models of Concurrency, June, Torino (Italy).

Berthomieu B.; Menasche M. (1983). *An Enumerative Approach for Analyzing Time Petri Nets*, IFIP Congress Series, Vol.9, p.41-46. Elsevier Science Publ. Comp. (North Holland).

Branicky M.S., Borkar V.S., Mitter S.K. (1998). *A unified framework for hybrid control : Model and optimal control theory*, IEEE Transactions on Automatic Control, Vol.43, N°1, p.31-45.

Buisson J., Cormerais H., Richard P.Y. (2001). *Formally computing the state equations for available configurations of bond graphs with switches*, ICBGM'2001, San Diego.

Champagnat R., Esteban P., Pingaud H., Valette R. (1998). *Modeling and simulation of a hybrid system through Pr-Tr PN-DAE model*, ADPM'98, p.131-137, Reims (France).

Demongodin I. (2001). *Generalised Batches Petri Net: Hybrid Model for High Speed Systems with Variable Delays*, Discrete Event Dynamic Systems: Theory and Applications, Vol.11, n°1/2, p.137-162.

Deshpande A., Gollü A., Semenzato L. (1998). *The shift programming language for dynamic networks of hybrid systems*, IEEE Trans. Automatic Control special issue on Hybrid Systems.

Fábián G., van Beek D.A., Rooda J.E. (1998). *Integration of the Discrete and the Continuous Behaviour in the Hybrid Chi Simulator*, European Simulation Multiconference, Manchester.

Gani R., Braunschweig B.L. (2002). *Software architectures and tools for computer aided process Engineering*, Elsevier, ISBN :0-444-50827-9.

Gear C.W. (1971). *The Simultaneous Numerical Solution of Differential-Algebraic Equations*, IEEE Transaction on Circuit Theory, CT 18 (1), Ed. Academic Press.

Hétreux G., Théry R., Perret J., LeLann J.M., Moyse A., (2002). *Bibliothèque orientée-objet pour la simulation dynamique des procédés : architecture et mise en oeuvre*, SIMO, Toulouse (France),

Hétreux G., Perret J., LeLann J.M. (2003). *Bibliothèque orientée objet pour la conception de simulateurs dynamiques hybrides*, Congrès Français de Génie des Procédés (CFGP'2003), 9-11 September, Saint-Nazaire (France).

Jourda L. (1996). *Composants logiciels Orientés Objets pour la modélisation et la simulation des procédés chimiques*, PhD thesis, INP, Toulouse (France)

Jourda L., Joulia X., Koehret B. (1996). *Introducing ATOM, the Applied Thermodynamic Object-Oriented Model*, Computer and Chemical Engineering, 20A, S157-S164.

Kesten Y. and Pnueli A. (1992) *Timed and hybrid statecharts and their textual representation*, Lecture Notes in Computer Science, Vol.N°571.

Le Bail J., Alla H. and David R. (1991). *Hybrid Petri Nets*, Proceedings of the European Control Conference, p.1472-1477, Grenoble (France).

Marquardt W. (1992), *An object oriented representation of structured process model*, ESCAPE 1

Moyse A. (2000). *Odysseo, plate-forme orientée-objet pour la simulation dynamique des procédés*. PhD thesis, INP, Toulouse, France.

Nilsson. (1993), *Object oriented modelling of chemical Processes*, Ph.D. thesis , Lund Institute of Technology,

OPERA, (1999) Operators Training Distributed Real-time Simulations, Esprit projet n°24950, Whitepaper.

Paludetto M., Valette R., Courvoisier M. (1990). *Génération de code Ada à partir d'une approche orientée objet Hood/Réseaux de Petri*, Journées Internationales Le Génie Logiciel et ses Applications, p.795-826, Toulouse (France).

Perret J. (2003). *Intégration des Réseaux de Petri Différentiels à Objets dans une plateforme de simulation dynamique hybride : application aux procédés industriels* , PhD thesis, INP, Toulouse (France).

Perret J., Hétreux G., Thery R., LeLann J.M. (2003). *Object-oriented components for dynamic simulation of a reactive distillation process*, European Symposium of Computer Aided Process Engineering (ESCAPE 13), 1-4 June, Lappeenranta (Finland).

Perret J., Hétreux G., LeLann J.M. (2003). *Object hybrid formalism for modelling and simulation of chemical processes.*, IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'03), 16-18 June, Saint-Malo (France).

Sargousse A. (1999). *Noyau numérique orienté-objet dédié à la simulation des systèmes dynamiques hybrides*, PhD thesis, INP, Toulouse (France).

Sibertin-Blanc C. (1985). *High-level Petri nets with Data struct*ure, 6th European workshop on Petri nets and applications, Espoo (Finland).

Valentin-Roubinet C. (1999), *Hybrid Systems modelling : Mixed Petri Nets*, 3rd IMACS/IEEE Conference CSCC 99, 4-8 Juillet, p.223-228, Athènes (Grèce).

Wöllhaf K., Fritz M., Schulz C., Engell S. (1996). BaSiP – *Batch Process Simulation With Dynamically Reconfigured Process Dynamics*, Supplement to Computers and Chemical Engineering, 20(972), p.1281-1286.

Zaytoon J. (2001). *Systèmes dynamiques hybrides*, HERMES Sciences publications.