AN OBJECT-ORIENTED APPROACH FOR HYBRID SYSTEM MODELLING

E. Villani*, P. E. Miyagi*, R. Valette+

* Escola Politécnica, University of São Paulo
Av. Prof. Mello Moraes, 2231 CEP 05508-900 São Paulo, BRAZIL
+ Laboratoire d’Analyse et d’Architecture des Systèmes – LAAS / CNRS
7, Avenue du Colonel Roche, 31077 Tolouse Cedex 4 FRANCE
e-mail: emiliav@usp.br, pemiyagi@usp.br, robert@laas.fr

Abstract: This paper presents a novel modelling approach for hybrid industrial plants. It is based on the introduction of object-oriented concepts to the Differential Predicate Transition nets. To handle the system complexity a top-down methodology is considered, where class models are successively refined and decomposed. The UML language is also used to represent different views of the modelled system. A cane sugar factory is used as an example to illustrate the proposed approach.

Keywords: object modelling techniques, Petri nets, differential equations, decomposition methods, top-down methods.

1. INTRODUCTION

One of the most important points for the automation of industrial production systems is the design of the supervisory system. The supervisory system is a real-time system that should concurrently react to aperiodic commands and continuously monitor state changes, according to the hierarchical control structure of Fig. 1 (Silva et al, 1998).

According to (Garcia, 1997), not rarely, the supervisory system is responsible for integrating local control systems of different nature, such as PIDs and PLCs, resulting in a increasing interest for hybrid systems, i.e., systems that have both discrete and continuous state variables (Alla & David, 1998).

In the development of supervisory system, the modelling approach is of particular importance. Furthermore, for the analysis of the supervisory system, an overall model is necessary that includes also the industrial plant and the local control system.

In this context, this paper considers the problem of hybrid plant and local control system modelling, while the modelling of the supervisory is presented in (Miyagi et al, 2001). It takes as a starting point a proposal of Petri net associated with differential algebraic equations (Differential Predicate/Transition nets) (Champagnat, 1998), to which it introduces the object-oriented concepts. The UML language (Booch et al, 1998) is also used to represent different views of the modelled system. Particularly, it approaches the problem of how to define objects and build their models for a complex hybrid plant. For this purposes a top-down methodology is used, where the system is successively decomposed and refined.

This paper is organised as following. Section 2 introduces the object-oriented concepts to the Differential Predicate Transition nets. In Section 3 the top-down modelling approach is presented using as an example a sugar cane factory. Finally, Section 4 draws some conclusions.
2. OBJECT ORIENTED APPROACH AND PREDICATE TRANSITION PETRI NETS

2.1 The Differential Predicate/Transition net (DPT net)

A number of formalisms have already been proposed for hybrid system modelling. Among them, those derived from Petri net are particularly considered in this work because of their well-know advantages of representing conflicts, causality, resource sharing and synchronisation (David & Alla, 1994). Among the hybrid formalisms derived from Petri net, the Differential Predicate/Transition net (DPT net) is chosen. Its main advantage is that it is the result of the merging of a discrete formalism (Petri net) with a continuous one (differential equation system). Therefore, it presents a broader modelling power, based on the class and object definition of Booch et al (1998), the proposed approach defines that:

- a set of formal variables (V) is defined for the net;
- each token is associated with a tuple of formal variables (X_i), which is a subset of V;
- each place (P_i) is associated with a differential equation system (F_i) that defines the value of the X_i associated to the tokens in P_i, according to the time (θ):

\[
F_i(X_i, X_i, θ) = \begin{cases} 
  f_{i-1}(X_i, θ) = 0 \\
  f_{i-n}(X_i, θ) = 0 
\end{cases}
\]  

(1)

- each transition (Tj) is associated with an enabling function (ε_j) that enables the transition firing according to the value of the X_i associated to the token of the input places (X_{input,i}):

\[
ε_j(X_{input,i}, θ) = 0
\]  

(2)

- each transition (Tj) is associated with a junction function (j_j) that defines the value X_i associated to the token of the output places (X_{output,i}) after the transition firing:

\[
X_{output,i}(θ) = j_j(X_{input,i}(θ))
\]  

(3)

2.2 The application of object oriented concepts to the DPT nets

Based on the class and object definition of Booch et al (1998), the proposed approach defines that:

- the behaviour of a class C is modelled by a DPT net (NC);
- the attributes of the class C is modelled as the set of variables of the PTD net (VC);
- the first variable of a token tuple of variables in a class net is the identity of an object;
- an object of a class is represented by a token in the class net, or by a set of tokens with the same identity.

2.3 Modelling the interaction between two objects

The communication among objects of different classes could be of two types: discrete and continuous.

The discrete type of communication is represented by method calls, which is modelled according to Paludetto (1991) and briefly illustrated in Fig. 2 for a synchronous method call:

![Fig. 2. Modelling of method call.](image)

The continuous type of communication is modelled by variable sharing among classes. Considering as X_{ext} a set of attributes of objects of other classes, the enabling function, the junction function and the differential equation system are then redefined as:

\[
e_j(X_{ext}, X_{input,i}, θ) = 0
\]  

(4)

\[
X_{output,i}(θ) = j_j(X_{ext}, X_{input,i}(θ))
\]  

(5)

\[
F_i(X_{ext}, X_i, X_i, θ) = \begin{cases} 
  f_{i-1}(X_{ext}, X_i, θ) = 0 \\
  \vdots \\
  f_{i-n}(X_{ext}, X_i, θ) = 0 
\end{cases}
\]  

(6)

It is important to observe that, in the differential equation system, X_{ext} acts as parameters used to solve the system and calculate the values of the variables of X_i. The value of X_{ext} is not determined by the DPT net of that class, but by the DPT net it comes from.

Based on the concept of object independence, each object should be able to determine its own time evolution, even if considering the interaction with other objects. This mean that each object should be able to calculate the value of its variables. The value of external variables must be just parameters.

An important consequence of this is that closed loop of variable sharing among objects are not allowed. A closed loop is the case when to compute the value of its variables, the object depends on the value of the variables of other object and vice-versa. As a result, the equations of the two objects should be solved as a single equation system.

![Fig. 3. Example of a NOT ALLOWED close loop.](image)
3. THE MODELLING APPROACH

3.1 The example

The case study considered is a cane sugar production process. A cane sugar factory consists of a series of mechanical and chemical processes grouped in stages. Some stages could be classified as batch processes, while others are essentially continuous. A diagram of the process is illustrated in Fig. 4.

When a cane truck arrives, a crane transports the cane to a conveyor (Reception Process). In the Preparation Process, the cane is chopped, shredded and sent to the milling train, which consists of four to seven mills. At each mill, cane juice is collected and the cane fibre continues to the next mill. The juice of the last mill is used to imbibe the cane of the precedent mill and so on. Only the juice of the first and second mill is collected and sent to the next stage, where it is mixed with SO₂ (to break the sucrose molecules into glucose molecules). To restore the pH of the juice it is then mixed with limewater. The juice is then pumped through a heater and sent to the Clarification Process. The clarifier is a continuous process equipment that consists of a number of compartments where particles settle. The resulted juice is sent to the five-stage evaporator to produce the cane syrup. The steam produced at each stage is used in the next stage. Furthermore, a part of the steam of the Evaporation Process is also used in the clarifier, in the heater and in the crystallisation pan. The syrup of the Evaporation Process is sent to the crystallisation pans, resulting in a sugar dough that is subsequently sent to the centrifuges.

3.2 The top-down modelling approach

For the cane sugar factory, a flat modelling approach, where all the classes are identified in a first moment, is unfeasible. A top-down modelling approach (Fig. 5) is therefore proposed.

In the top-down modelling approach, an initial set of classes is defined and modelled using DPT nets. Each class model is then decomposed into a new set of class. These class models are then refined, augmenting their level of detail, and subsequently decomposed. The activities of decomposing and refining should continue until the model reaches the desirable level of details.

Throughout this process, the UML diagrams are used to represent different views of the system, highlighting key aspects that characterises the evolution of the modelling process.

3.3 The initial set of classes

In an industrial plant, the more likely candidates to classes are the processed entities, such as products and raw material, and the resources, such as equipment. This is due to the fact that, whenever possible, classes of a model should have a strict relation with the real entities of the system.

In the first category, for the cane sugar factory, the main candidates are the \( \text{CANE} \) and the \( \text{CRYSTAL SUGAR} \). The \( \text{CANE} \) class represents the cane in truck, before it being put in the conveyor.

For the second category, the \( \text{PLANT EQUIPMENT} \) could be considered as a first class. However, it is reasonable that, before a first attempt of modelling with DPT net, this candidate should be decomposed.

The decomposition of a class should be made considering that, after the decomposition, each new class should have a strong internal cohesion and a weak connection with the other new classes (Paludetto, 1991). Based on this, the following rule are considered for the class decomposition:

1st Rule - The new classes resulted from the decomposition could have only discrete interaction among them.

The equipment related to each process of Fig. 4 is considered as possible candidates to new classes resulted from the \( \text{PLANT EQUIPMENT} \) decomposition (\( \text{RECEPTION EQUIPMENT}, \text{ PREPARATION EQUIPMENT}, \text{ MILLING EQUIPMENT}, \text{ SULPHATE ADDITION EQUIPMENT}, \text{ etc.} \)).

Analysing the process interactions, among the set of candidates, the \( \text{RECEPTION EQUIPMENT} \) is a valid one because it interacts with the other candidates by the discrete events of taking the cane from the truck and putting it into the conveyor. However, all the next processes until the crystallisation involve continuous interactions.
interactions represented by the flow of cane, cane juice and cane syrup. Finally, the interactions of CENTRIFUGATION EQUIPMENT could be considered as a valid candidate because the sugar dough loading and unloading are discrete events.

As a result the initial class PLANT EQUIPMENT is decomposed into three classes: RECEPTION EQUIPMENT, CENTRIFUGATION EQUIPMENT and CONTINUOUS FLOW EQUIPMENT (which includes the equipment of all the other processes). The CONTINUOUS FLOW EQUIPMENT is still a too complex object to be modelled as a DPT net. Therefore a second rule is introduced for its further decomposition.

2nd Rule - The new classes resulted from the decomposition could have continuous interaction but restricted to time intervals.

Not considering the situation when all the plant equipment is off (naturally, in this situation there is no continuous interaction), the flow through the preparation, milling, sulphur and stock equipment is continuous. One process could not be interrupted without interrupting also the others. So the interaction among the related possible classes is not restricted to a time interval. On the other hand, the continuous interaction of the lime addition equipment with both stocks is restricted to the time interval of loading and unloading the tank. Following, the flow through the stock tank, heater, clarifier and evaporator is also continuous. Lastly, the continuous interaction of the crystallisation equipment is restricted to the time interval of loading the crystallisation pan. As a result the initial object CONTINUOUS FLOW EQUIPMENT is decomposed into four objects: JUICE EQUIPMENT, LIME ADDITION EQUIPMENT, SYRUP EQUIPMENT and CRYSTALLISATION EQUIPMENT.

The definition of the initial set of objects described above is represented in Fig. 6 by the UML Class Diagram. The relation between the classes resulted from the decomposition and the initial class is UML relationship of composition (represented by a filled diamond). The classes represented by a rectangle with a continuous line belong to the initial set of objects that will be modelled with DPT nets. The classes represented by a rectangle with a dashed line are intermediate classes and will not be modelled.

Because of the limited number of pages, only examples of models for the RECEPTION EQUIPMENT (Fig. 7), CANE (Fig. 8), and JUICE EQUIPMENT (Fig. 7) classes are presented.

Fig. 7. RECEPTION EQUIPMENT Model.

Fig. 8. CANE Model.

Fig. 9. Juice Equipment Model.

Token variables:
\[ X = \{ a, h_{\text{in}}, h_{\text{out}}, f_{\text{in}}, f_{\text{out}}, \text{Vol} \} \]

Equation systems:
Place OFF:
\[ \text{Vol} + f_{\text{out}} = 0 \]
Place ON:
\[ \dot{a} - 1 = 0 \]
\[ h_{\text{in}} - \exp\left[ - (a - K_1)^4 - (a - K_2)^2 \right] - K_3 = 0 \]
\[ h_{\text{out}} - \frac{K_4 \cdot h_{\text{in}} - K_5 - h_{\text{out}}}{\tau_1} = 0 \]
\[ f_{\text{in}} - \frac{K_6 \cdot h_{\text{out}} \cdot n - f_{\text{in}}}{\tau_2} = 0 \]
\[ \text{Vol} - f_{\text{in}} + f_{\text{out}} = 0 \]

Enabling functions:
\[ e_1; \text{Vol} = 0 \quad e_2; \text{Vol} = 0 \]

Junction function:
\[ J_1: f_{\text{out}}(\theta^*) = K_7 \quad J_2: f_{\text{out}}(\theta^*) = 0 \]
\[ J_3: f_{\text{out}}(\theta^*) = 0 \quad J_4: f_{\text{out}}(\theta^*) = K_7 \]
\[ J_5: f_{\text{out}}(\theta^*) = 0 \quad J_6: f_{\text{out}}(\theta^*) = 0 \]
\[ J_2: f_{\text{out}}(\theta^*) = 0 \quad J_7: a(\theta^*) = 0 \]

3.4 Class modelling with DPT net

Each class should be modelled using the DPT net.
v – volume of cane in the truck;
a – the time since the last unloading;
h_{in}, h_{out} – height of the cane layer in the beginning and at the end of the conveyor;
f_{in} – flow of juice cane entering in the Stock tank;
f_{out} – flow of juice cane leaving the Stock tank;
Vol – volume of juice in the Stock tank;
K, K_1, K_2, K_3, K_4, K_5, K_7 – model constants;
\tau_1, \tau_2 – time delay constants.

3.5 Discrete class interfaces modelling

For the definition of the class interfaces, first the possible use cases should be identified. For each use case, the sequence of events should be modelled using the Collaboration Diagram of UML. An example is illustrated in Fig. 10 for the use case of arrival of a new truck (for not overcharging the figure only the first 9 message calls are represented). In this figure, the LOCAL CONTROL SYSTEM is an external element that interacts with the plant model.

Based on the Collaboration Diagrams, which identifies the methods of each class, the methods interface should be added to the class models. Whenever necessary, the class models should be revised and adapted to consider new states and events necessary to the method implementations. The interactions among classes are also modelled. An example is presented in Fig. 11.

3.6 Class model decomposition

Once the model of each class has been built, it could be further decomposed considering the relation among its part. As an example Fig. 12 presents the decomposition of class RECEPTION EQUIPMENT into two new classes: MOVEMENT SYSTEM and CLAW. After this decomposition, the collaboration diagram should be updated and the interaction among the two new classes should be added to the class models.

For the decomposition of the JUICE EQUIPMENT class, a third rule is defined:

3rd Rule - The new classes resulted from the decomposition could have continuous interaction not restricted to time intervals but could not allow closed loop sharing of variables.

Observing the equation system of Fig. 9, the JUICE EQUIPMENT decomposition is proposed in Fig. 13:

Token variables:
X={f_{in}, f_{out}, Vol}

Equation systems:
Place ON:
\dot{a} = 1 = 0
h_{in} - \exp[-(a - K_1)^4 - (a - K_2)^2] - K_3 = 0
\dot{h}_{out} - \frac{K_4 * h_{in} - K_5 - h_{out}}{\tau_1} = 0

Token variables:
X={f_{in}, f_{out}, Vol}
relations and sequence of method calls, and helping on the interface definition, the UML diagrams are used.

At this moment, techniques for the model analysis are being studied. Beforehand, some consideration could be made. Due to the fact that the model includes real variables, the number of possible states is unlimited. Therefore, no method could be proposed for the verification of the net properties (such as liveness, boundness, etc.) considering the hybrid nature of the model. However, the formal analysis could be carried out for the ordinary Petri net model of each class (or for the DPT net, according to its complexity). Initially the model of each class should be analysed, then, by the successive merging of classes (considering the decomposition hierarchy) and simplification of the resulting model, the overall system model could also be analysed. As the DPT nets do not impose any restriction on the differential equation system (it could be linear, non-linear, etc.), it could not be guaranteed that the differential equation system has an algebraic solution. Therefore, the continuous part could only be integrated in the analysis by simulation.

ACKNOWLEDGES
The authors would like to thank the partial financial support of the governmental agencies FAPESP, CNPq, CAPES and RECOPE/FINEP.

REFERENCES