An algorithm for the evolution graph of extended Hybrid Petri nets [*]

Martina Svadova
Center for Applied Cybernetics
Czech Technical University
Prague 2, Czech Republic
xsvadova@control.felk.cvut.cz

Zdenek Hanzalek
Center for Applied Cybernetics,
Czech Technical University
Prague 2, Czech Republic
hanzalek@control.felk.cvut.cz

Abstract - *Extended Hybrid Petri nets (eHPNs) defined by David & Caramihai are one of possible extensions of Hybrid Petri nets modeling a delay on continuous flow. The behavior of hybrid dynamic systems, modeled by eHPNs, can be studied using an evolution graph. This paper introduces an algorithm generating the evolution graph consisting of IB-states. A model of hydro-system is used as an illustrative example.*

Keywords: Delay, Extended Hybrid Petri nets, Evolution graph algotihm

1 Introduction

Petri nets (PNs) [1],[6] are generally used to model and analyze discrete event dynamic systems like manufacturing systems and communication protocols. Hybrid systems consisting of discrete and continuous parts can be modeled by Hybrid Petri nets [2],[5].

First-Order HPNs (FOHPNs) were studied in [7] by Balduzzi, Guia and Seatzu. In contrast to classical HPNs defined by David & Alla this type of PNs has two main differences in continuous part. First, in FOHPN continuous transitions are always strongly enabled. Second, instantaneous firing speed can be constrained by minimum firing speeds in FOHPN.

Neither the classical Hybrid Petri nets [5] nor the FOHPN do not allow to model the systems with delay on continuous flow (e.g. the product delay on the conveyor, the delay of fluid in pipe). Extended Hybrid Petri nets (eHPNs) defined by David & Caramihai [3] are one of possible extensions of Hybrid Petri nets modeling delay on continuous flow. This article aims at algorithmisation of this model.

Behavior of deterministic hybrid dynamic systems, modeled by eHPNs, can be analyzed by an evolution graph, which is composed from so-called invariant behavior states (IB-state) and transitions between particular IB-states.

This paper describes an algorithm generating an evolution graph consisting of IB-states characterized by constant marking of discrete places, by constant instantaneous firing speed of continuous transitions. Further the IB-state is characterized at its entry point by the continuous marking and by the elapsed time of each enabled discrete transition. The transitions between IB-states are characterized by the event provoking the transition and by the elapsed time in previous IB-state. The algorithm is part of the PN Matlab Toolbox [4].

The paper is organized as follows: Section 2 briefly presents eHPNs. Section 3 describes the algorithm for generating of the eHPNs. Section 4 presents an example showing functionality of the presented algorithm. A model of a hydro-system is used as illustrative example.

2 Extended Hybrid Petri nets

Basic notion of an extended Hybrid Petri nets (eHPNs) is defined in this section. The extended Hybrid Petri nets defined by David & Caramihai [3] allow to model systems with delays on the continuous flow. They are extension of classical Hybrid Petri nets defined by Alla & David [2] and they assume all their properties.

A marked timed eHPN is defined by the seven-tuple:

$$\langle \mathbf{P}, \mathbf{T}, \mathrm{Pre}, \mathrm{Post}, M_0, V, d \rangle \quad (1)$$

$\mathbf{P} = P \cup CP$	finite and non-empty set of places, (P - the set of discrete places and CP - the set of continuous places (P and CP are disjoint)
$\mathbf{T} = T \cup CT \cup eT$	finite and non-empty set of transitions, where T is the set of discrete transitions; eT is the set of extended transitions and CT is the set of continuous transitions (T, CT and eT are disjoint)

[*] **0-7803-8566-7/04/$20.00 © 2004 IEEE.**

Pre: $\mathbf{P} \times \mathbf{T} \rightarrow N$ or $R^+ \cup 0+$	input incidence matrix
Post: $\mathbf{P} \times \mathbf{T} \rightarrow N$ or $R^+ \cup 0+$	output incidence matrix
$M_{0:}$ $\mathbf{P} \rightarrow N$ or $R^+ \cup 0+$	vector of initial marking
$V=\{eV_1,.,eV_m;V_1,.,V_m\}$	vector of maximal speeds; speed V_j is associated with continuous transition CT_j and speed eV_j is associated with extended transition eT_j
$d=\{ed_1,.,ed_n;d_1,.,d_n\}$	vector of time delays, which expresses firing delay of transition; time delay d_j is associated with discrete transitions T_j and ed_j is associated with extended transitions eT_j

Extensions of eHPNs compared to HPNs :

- the weight of an arc, leading from or leading to continuous place, may be infinitely small positive real number, i.e. 0+
- the marking of continuous place can be $R^+ \cup 0+$ instead of $\mathcal{R}^+$
- presence of inhibitor arcs (not assumed in this article)

The extended transition in eHPN ensures delays on the continuous flow. The extended transition can have only one input and one output place which is the continuous one. The output and input continuous place is connected with extended transition by arcs with weight 0+. As well as in the case of discrete transition, the time delay is associated to extended transition. The weight 0+ of an arc allows continuous firing of the extended transition the time delay is elapsed. Symbol 0+ represents infinitely small positive real number. The marking of the input place of the extended transition can contain batches of marks, where each batch has different time elapsed.

2.1 The evolution graph

An evolution graph of eHPN is an oriented graph consisting of nodes corresponding to IB-states (invariant behavior states) and transitions between these nodes. The IB-state characterizes the state of eHPN during certain time. The marking of discrete places and the instantaneous firing speed of continuous transitions remain constant as long as the system is in the same IB-state. The transition of the evolution graph corresponds to an event provoking passage from one IB-state to another IB-state. It is labeled by the name of the events and by the time elapsed in previous IB-state (eventually by the absolute time).

An IB-state consists of two parts in eHPNs (see Figure 1):

- the discrete part on the left side of the IB-state is given by M_D (the marking of discrete places) and by D, the elapsed time of each enabled discrete transition at the beginning of the IB-state.

- the continuous part on the right side of the IB-state is given by v_E, the instantaneous speed of extended transitions, by v_C, the instantaneous firing speed of continuous transitions and by M_C, the marking of continuous places at the beginning of the IB-state.

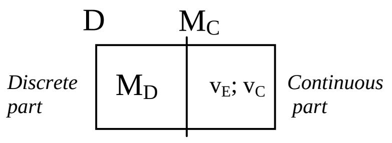


Figure 1 - The structure of IB-state

In eHPNs, a passage from one IB-state to the following one can occur only if any of the following of events occurs:

D-event:	a discrete transition is fired
C-event:	the marking of a continuous place with negative balance becomes 0. The balance, the derivative of the continuous place marking, is constant in a given IB-state. The balance is calculated for each continuous place and determines dynamics of the place [1]
A-event:	the marking of a continuous place, whose balance is positive and which is input place of discrete transition, reaches weight of arc linking this place with the discrete transition
E-event:	the continuous firing of a discrete transition either begins or ends

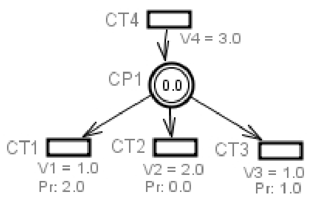
3 Algorithm

An algorithm for construction of the evolution graph of extended Hybrid Petri nets is presented in next paragraphs. The algorithm is designed to simulate also behavior of systems modeled by PN, CPN and HPN. The algorithm works with single server semantics (a firing delay is set, when the transition is enabled and new delay is generated upon transition firing only if the transition is still enabled in a new marking) and it does not work with unbounded nets. The presented algorithm solves conflicts in continuous part of the net without the loop of continuous transitions. Notion “conflict in continuous part” supposes that continuous place has at least two output continuous transitions, which are weakly enabled and have not any output discrete transition. The conflict of continuous place is solved by priority assignment to output continuous transitions. It means the input stream of the place (the sum of speeds of all CTj such that CTj $\in$°CPi) is divided among the transitions up to their priority, starting with the highest priority (corresponding to the highest value). The instantaneous firing speed of weakly enabled continuous transitions is calculated by iterative algorithm 3.2 presented in [1], where equation (2) was extended by vector Infl. The algorithm runs until speed vector is equal to the one from previous iteration, i.e. $v^{r+1} = v^r$ (r is index of iteration in the algorithm).

$$v_j^{r+1} = \min(V_j, (\sum_{k=1}^{j-1}(Post(i,k) - Pre(i,k) * Infl_j) * v_k^{r+1} + \sum_{k=j+1}^{t}(Post(i,k) - Pre(i,k) * Infl_j) * v_k^{r})) \qquad (2)$$

The vector $Infl_j$ determines an influence of transitions on IFS calculation of transition CTj, therefore it is unique vector for each currently calculated transition. The number of Boolean valued entries in the vector corresponds to the number of continuous and extended transitions. Since by definition extended transitions cannot participate in a conflict, the elements of the vector corresponding to extended transitions are set to 0. Values of elements are set up to the priority of the calculated transition. The entries corresponding to the transitions with higher priority have value 1 and the entries corresponding to the transitions with lower priority have value 0. When the transition CTj is not in the conflict, all entries of vector $Infl_j$ are set to 1.

Example



$$v_1 = \min(V_1, v_4)$$
$$v_2 = \min(V_2, v_4 - v_1 - v_3)$$
$$v_3 = \min(V_3, v_4 - v_1)$$

$$\boldsymbol{Infl_1 = (}0,0,0,0\boldsymbol{)}$$
$$\boldsymbol{Infl_2 = (}1,0,1,0\boldsymbol{)}$$
$$\boldsymbol{Infl_3 = (}1,0,0,0\boldsymbol{)}$$

Figure 2- conflict in continuous part of Petri net

Figure 2 illustrates continuous transitions in conflict, calculation of their IFS by (2) and the values of vectors Infl. Transition CT1 has maximal speed V1=2 and priority 2, transition CT2 has V2=1 and priority 0 and transition CT3 has V3=1 and priority 1. All of them are weakly enabled. The CP1 input stream is equal to the maximal speed of CT4 (V4=3).

The model adopted in this text considers that a discrete transition does not reserve tokens in its input places. The case showed in Figure 3, where discrete transition T2 is the output transition of discrete place P1 and simultaneously there is self-loop formed by discrete place P1 with continuous transition CT1, is allowed and it not considered as a conflict. Since the reservation of tokens is not considered, transition CT1 can share marking of P1 until the discrete transition is fired.

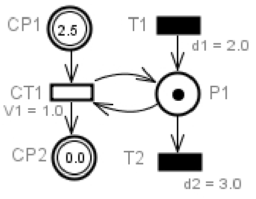


Figure 3 – Continuous transition and discrete transition sharing one discrete place

3.1 The principle of the algorithm

The principle of the algorithm is based on finding of events evoking a passage from one IB-state to the following one. First for given initial marking, the times, in which particular events occur, are found. The times acquired in previous step are compared and the minimal time is found. In case that more times have the same value as the minimal time, corresponding events are merged to vector of events associated with the minimal time. The new marking and new timing vector are generated at this time. Assuming bounded net, this procedure is repeated until a terminal node represented by stable speed state is reached or the net gets into deadlock (type of terminal node) or the loop of IB states is found.The block diagram of the algorithm is illustrated in

Figure 4.

3.2 Detailed description of the algorithm:

Step 1. Variables initialization; tests of validity of a net structure

The algorithm is designed to be applicable to simulate behavior of systems modeled by PN, CPN and HPN. In this step kind of a net is identified, variables by kind of the net are initialized and validity of the net is tested.

Step 2. Particular calculations by kind of the net

Following calculations are divided up to kind of the net to three parts: continuous, discrete and extended.

Continuous part:

Enabled transitions of continuous part are calculated by an iterative algorithm 3.1. presented in [1]. Instantaneous firing speeds (IFS) of weakly enabled continuous transitions are calculated by algorithm 3.2. also presented in [1].

Discrete part:

Enabled transitions of discrete part meet a condition $M(P_i) \geq Pre(P_i, T_j)$ for all places in $°T_j$ (set of input places of transition T_j). Minimal time of D-event is found by comparing actual timings associated with enabled discrete transitions.

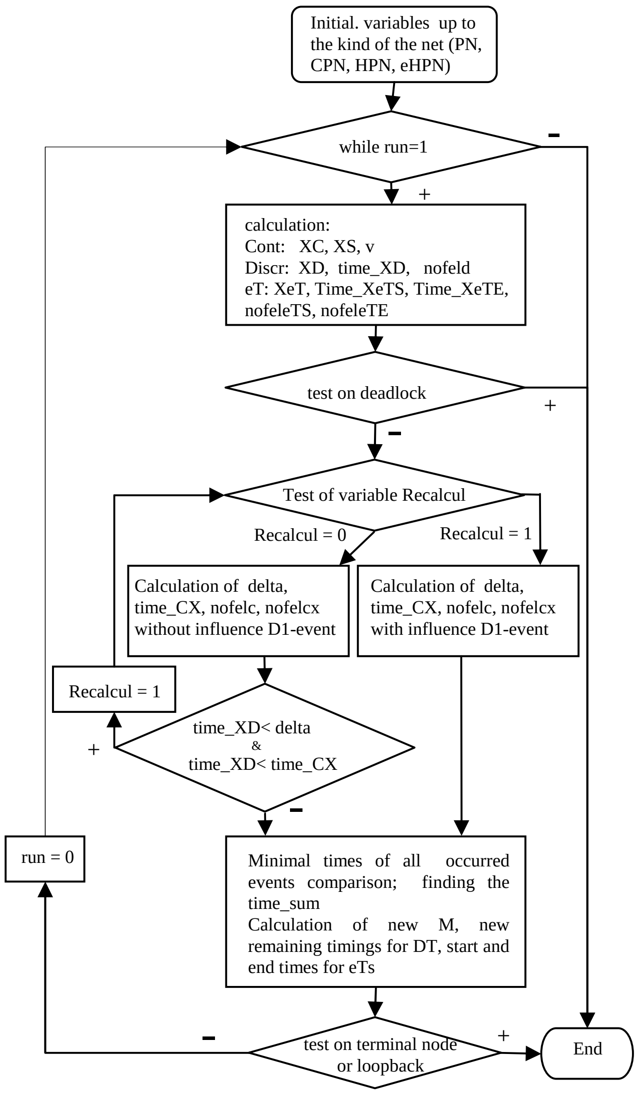

Initial. variables up to the kind of the net (PN, CPN, HPN, eHPN)
while run=1
-
+
calculation:
Cont: XC, XS, v
Discr: XD, time_XD, nofeld
eT: XeT, Time_XeTS, Time_XeTE, nofeleTS, nofeleTE
test on deadlock
+
-
Test of variable Recalcul
Recalcul = 0
Recalcul = 1
Calculation of delta, time_CX, nofelc, nofelcx without influence D1-event
Calculation of delta, time_CX, nofelc, nofelcx with influence D1-event
Recalcul = 1
time_XD< delta & time_XD< time_CX
+
-
run = 0
Minimal times of all occurred events comparison; finding the time_sum
Calculation of new M, new remaining timings for DT, start and end times for eTs
-
test on terminal node or loopback
+
End

Figure 4 – The block diagram of the algorithm

Extended part:

Extended transition eTj is enabled if it meets conditions $M(P_i) \geq Pre(P_i, T_j)$ for the place in $°T_j$ or input stream of $°T_j$ is nonzero. Minimal times of extended event are found by comparing actual timings associated with enabled discrete transitions.

Step 3. Deadlock testing

If the vector of enabled transitions is equal to zero, the algorithm terminates. If not continue by *Step 4*.

Step 4. Variable *Recalcul* testing; Calculation of minimal times of C-event and A-event:

C-event: Markings of places, which have negative balance, are divided by particular value of negative balance. The minimal time is found by comparing results of this operation.

A-event: Marking of each place, which has positive balance and which is input place of discrete transition and whose previous marking was smaller than weight of arcs linking this place with discrete transition, is divided by particular value of positive balance. The minimal time is found by comparing results of this operation.

A variable *Recalcul*, which determines if balance vector (C and A-event are calculated from balance vector) is calculated with influence of the discrete transition, is tested. In the case when *Recalcul* is equal to 0, minimal times of C-events are calculated without influence of discrete part (all elements of vector Xdfir, determining firing of discrete transitions, are equal to 0), then continue by *Step 5*. In the case when *Recalcul* is equal to 1, times of C-event and A-event are calculated with affect of discrete part (elements corresponding to fired discrete transitions are equal to 1), then continue by *Step 6*.

Step 5. Test, if discrete part of HPNs affects continuous part

The times of *C-event* and *A-event* calculated in previous step are compared with minimal time of *D-event*. If time of *D-event* is greater than times of *C-event* or *A-event* (in this case *D-event* does not affect continuous part), continue by *Step 6.*

If the minimal time of *D-event* is equal to minimal times of the *C-event* or *A-event,* D-event affects continuous part, and particular times of *C-event* and *A-event* have to be re-calculated with influence of *D-event* (balance is calculated with influence of D-event). Since at this time the new events of *C-event* and *A-event* can occur, they have to be also related to corresponding transition in the evolution graph (in this case there are several kinds of events at the same time). Variable *Recalcul* becomes 1. Continue by *Step 4*

The reason of recalculation C and A events, when D-event influences them:

Figure 5 illustrates the case when calculation of minimal time of C-event is influenced by discrete part. In time t=0s, the transition T1 becomes enabled and its timing is set to 2. Simultaneously the IFS of transition CT1 (v1) is equal to 1. If C-event (the marking of CP1 becomes 0) is calculated without effect of a discrete part, C-event would occur in time t=2s. Since in time t=2s D-event also occurs (transition T1 is fired), marking of place CP1 becomes 1. C-event does not occur in time t=2s, but in time t=3s, and therefore C-event has to be recalculated with effect of discrete transition T1.

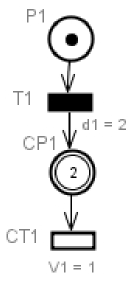


Figure 5 – Influence of D-event on C-event

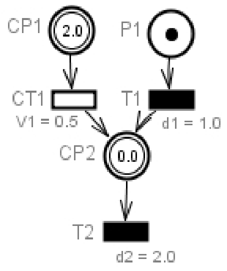


Figure 6 – Influence of D-event on A-event

Figure 6 illustrates the case when calculation of minimal time of A-event is influenced by discrete part. In time t=0, the transition T1 becomes enabled and its timing is set to 1. Simultaneously IFS of transition CT1 (v1) is equal to 0.5. Without the influence of discrete part, A-event would occur in time t=2s. However in time t=1s transition T1 is fired and the marking of continuous place CP1 becomes 1.5. Discrete part of the net influences occurrence of A-event. By this A-event does not occur in time t=2s, but earlier and therefore C-event has to be recalculated with effect of discrete part.

Step 6. *Selection of events with total minimal time; generation of the event vector; calculation of new marking corresponding to occurred event/events; detection of the loopback*

The total minimal time is found by comparing minimal times of particular kinds of events found in previous steps. All events, which are related to total minimal time, are merged to vector of events. The new initial marking is calculated, the time elapsed in previous IB-state and new timing vector are generated. Evolution of behavior of the net can be stable, terminated by terminal node, or cyclic, represented by loop. The behavior is cyclic, when marking of new IB-state is equal to the marking of existing i-th IB-state and simultaneously timing vector is equal to i-th column vector in matrix of timings (i.e. the evolution graph forms a loop).

If the new marking is equal to the marking of previous state and timing vector as well (no evolution in the discrete part and stable instantaneous firing speeds in the continuous part) then the terminal node is reached. When net gets into deadlock, the last IB-state is also supposed as terminal node. If terminal node or cyclic behavior is found, the program terminates otherwise it continues by *Step 1.*

4 Example: Model of hydro-system

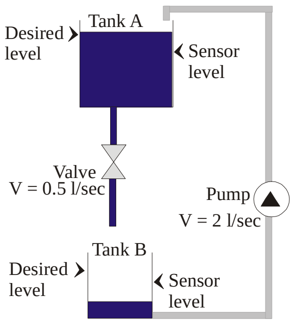


Figure 7 – Hydro-system

Figure 7 shows a hydro-system consisting of two tanks (tank A, tank B), one valve and one water pump. Each tank has Boolean valued sensor of water level. Since both the pump and the valve work with time delay, the levels of both tanks need to be modified. Desired level of tank A is 90 liters, but the sensor should detect 80 liters due to the time delay of the valve (5s ~ 10 liters of water). Desired level of tank B is 45 liters, but the sensor should detect 40 liters due to the time delay of the pump (10s ~ 5 liters of water). If the valve is open, the water from the tank A flows to the tank B by gravity at speed 0.5 l/sec. When the valve is open, the water pump does not pump and vice versa. Water is pumped from tank B to tank A by the pump at speed 2 l/sec. When pump stops pumping water in pipe flow back to tank B by gravidity at speed 0.5 l/sec. Model takes in account transfer delay of water. In the model there are two transfer delays modeled by extended transitions. Transfer delay water of the pipe from tank A to tank B is 4s. Transfer delay of the pipe from tank B to tank B is 5s. In the initial state, tank A contains 100 liters of water, tank B contains no water, the pump is off and the valve is open.

Figure 8 shows extended hybrid Petri net model of hydro-system in Figure 7. Place CP1 represents tank A, place CP2 represents tank B. Transition CT1 represents the valve and transition CT3 represents the pump. Transfer delay of the pipe from tank A to tank B is represented by transitions CT3, CT4 and eT2 and by places CP6, CP7 and CP8. Transfer delay of the pipe from tank B to tank A is represented by transitions CT1, CT2 and eT1 and places CP3, CP4 and CP5. Transition CT5 represents discharge of water to tank B when the pump stops pumping. Discrete part represents the control of hydro-system when P1 is marked, the pump is on and valve is closed; when P2 is marked, the valve is open and the pump is off. Firing of transition T1 evokes opening of the valve and stopping of the pump. Firing of transition T2 evokes closing of the valve and start of pumping.

Figure 9 shows evolution graph, which represents behavior of hydro-system. Evolution graph is constructed from output parameters of matlab function eHPN.m.

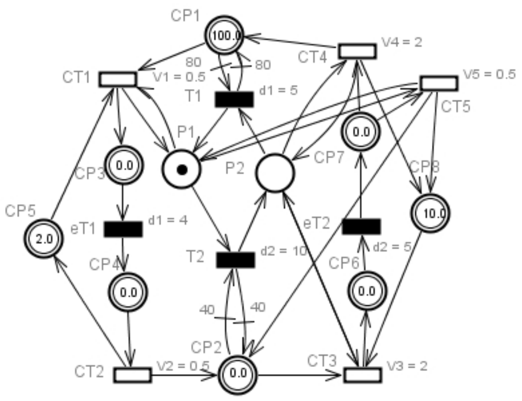


Figure 8 – eHPN model of hydro-system in Figure 7

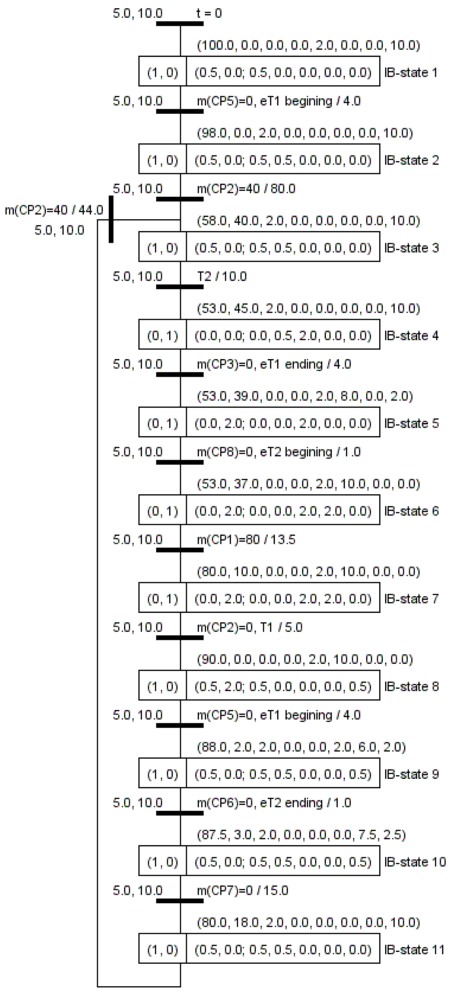


Figure 9 – evolution graph

5 Conclusion

In this paper we have shown the algorithm generating the evolution graph for extended Hybrid Petri nets with firing delays associated with discrete transitions, maximal speeds associated with continuous transitions and with continuous firing of extended transitions. This algorithm also solves conflict in continuous part of eHPNs and can be used for simulation of hybrid systems with continuous product flow. The algorithm is implemented as matlab function of the Matlab Toolbox for Petri nets and it is available from authors upon request.

This work supported by the Ministry of Education of the Czech Republic under Project LN00B096.

6 Reference

[1] R. David, and H. Alla, *Petri Nets and Grafcet,* Prentise Hall, London, 1992

[2] R. David, and H. Alla, *"Continuous and Hybrid Nets"*, *Journal of Circuits, Systems and Computers,* Vol 8, No. 1, pp. 159-188, 1998.

[3] R. David, and S.I. Caramihai, *"Modeling of Delays on Continuous flows Thanks to Extended Hybrid Petri nets"*. The 4th International conference on Automation of Mixed Processes: Hybrid Dynamic systems, p. 343-350, 2000.

[4] Z. Hanzalek, and M. Svadova, *"Matlab Toolbox for Petri Nets. Tool Demonstrations"*, Application and Theory Petri nets, Newcastle upon Tyne, p. 35-39, June 2001

[5] R. David, and H. Alla, *"On Hybrid Petri Net"*, *Discrete event dynamic systems: Theory and applications*, Vol 11, p. 9-41, 2001.

[6] T. Murata, *"Petri Nets: Properties, Analysis and Application",* Proceeding of The IEEE, Vol 77, 1989

[7] F. Balduzzi, A. Guia, and C. Seatzu, *"Modelling manufacturing systems with First-Order Hybrid Petri Nets"*, *International Journal of Production Research*, Special Issue on *Modeling, Specification and Analysis of Manufacturing Systems*, Vol. 39, No. 2, 2001