7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

TIMED HYBRID PETRI NETS VISUAL SIMULATION IN VPNP

Emilian GUȚULEAC[1], Vasile GÎSCA[2], Ludmila GUȚULEAC[3], Sergiu ZAPOROJAN[4]
Technical University of Moldova, Computer Science Department,
Bd. Ştefan cel Mare, nr. 168, MD-2012, Chişinău, Moldova
[1)] egutuleac@mail.utm.md, [2)] gasca@mail.utm.md, [4)] s_zaporojan@mail.utm.md

Abstract. *This paper presents a visual simulation of timed hybrid Petri nets (HPN) a class of Petri nets in which some places may hold fluid rather than discrete tokens. The simulation software has been developed and implemented in Visual Petri Nets + tool (VPNP) and offers a graphical user interface. The continuous transitions of the net are fired with speeds that are piecewise constants or variable.*
Keywords*: simulation, timed hybrid Petri nets, performance evaluation, software tool.*

1. Introduction

Researches in the aria of specification, modeling and simulation of parallel hybrid systems aim at verification, coherence and speed up of systems development, starting from requirement capture, high level and low level specification up to final prototyping. These goals are achieved through verification, translation and refinement of formal models. They can be applied to large dynamic information and computer systems. Specification as well as structural and quantitative analysis of hybrid systems have gained much attention over the last years. Formal modeling and simulation is a key for verification of hybrid systems specifications with discrete and continuous variables. Discrete variables may represent the states of a switching or controlling network or changing environment sates; the continuous variables usually represent physical quantities such temperature, molecular concentration or electrical potential. In some case, hybrid systems are characterized by having non-trivial between the discrete and continuous part of model; in this case a modeling environment is desirable which allows a unified specification of the complete system to be studied. Moreover, we would like the modeling paradigm to be convenient, flexible and intuitive. Among the formalisms that are used, Petri nets (*PN*) are the most popular. A number of different classes of *PN* have been proposed [1, 2, 7, 9]. Hybrid Petri nets (*HPN*) are *PN* based models in which some places may hold a discrete number of tokens and places a continuous quantity represented by a real number. *HPN* were introduced in [1], and in [3] were extended to include non-determinism both for firing times and rate of continuous transitions. Enabled continuous and discrete transitions may then fire according to their firing speeds or time delays, respectively. Timed *HPN* (*THPN*) aim at providing an approximation to discrete-state systems in which the number of objects (customers, packets, tasks, etc.) becomes exceedingly large to be treated with the discrete state approach common to timed *PN*.

HPN structure is formed by three kinds of discrete and continuous objects: places, transitions and arcs. Places represent some kind of resources, transitions denote actions or events that happen in the system (i.e. production of new recourses using existing) and arcs link the first two kinds of objects together, implementing the logic of the *HPN*; they assign actions to resources, and vice versa. Certainly this is only a basic set of objects. Nowadays scientists have developed several very useful extensions for *HPN*, but the basics stay the same. It may sound simple, but when it comes to building models using things it becomes little more difficult. It is because Petri nets are intuitive, but not simple in any way. Though their structures are simple, analysis of *HPN* is very complicated thing. There are several groups all over the world working on analysis of Petri nets, and certainly there is a bunch of modelling software packages [4, 5, 8, 10]. Most tools provide a graphical user

interface for the convenient drawing and editing of the model. They usually include analysis components for one or several classes of models. But still there is not such a program that would satisfy everyone's needs, both commercial and educational, so we decided to start developing our own software for modelling using Petri nets, and called it Visual Petri Nets + (*VPNP*).

The paper is organized as follows. In Section 2 we describe the model definition of extended timed *HPN*. Subsequently in Section 3 we considered the *VPNP* tool overview. Visual simulation facility of timed *HPN* in *VPNP* is presented in Section 4.Conclusions are drawn in Section 5.

2. Model definition of timed HPN

Various extensions have been made to the framework of timed *HPN* since its introduction in [1, 2, 6, 7]. The definition of extended *HPN* with customary notation is derived from [4]. We define an timed extended *HPN* structure as a 13-tuple $HSPN = < P, T, Pre, Post, Test, Inh, M_0, B, G, Pri, \theta, W, V >$, where:

- P is the finite set of places partitioned into a set of discrete places P_D, and a set of fluid places P_C, with $P=P_D \cup P_C$, $P_D \cap P_C=\varnothing$. The discrete places may contain a natural number of tokens, while the marking of a fluid place is a non-negative real number (fluid level). In the graphical representation, a discrete place is drawn as a single circle while a fluid place is drawn with two concentric circles;
- T is a finite set of transitions that can be partitioned into a set T_D of discrete transitions and a set T_C of continuous transition, $T = T_D \cup T_C$, $T_D \cap T_C = \varnothing$. A discrete transition $t_j \in T_D$ is drawn as a black bar and continuous transition $t_i \in T_C$ is drawn as a rectangle;
- $\forall p \in P_D$ and $\forall p \in P_C$, *Pre*, *Post*, *Test* and $Inh: P \times T \to Bag(P)$ is respectively a forward, backward, test and inhibition applications functions for the multi-sets of P, where defined the set of arcs A and describes the marking-dependent cardinality or continuous set of the *input* and *output discrete* (*flow reset*) arcs connecting transitions and places. The set of arcs A i*s* partitioned into five subsets: A_d, A_s, A_h, A_c and A_t. The subset A_d contains the *discrete normal arcs* which can be seen as a function A_d: $((P_D \text{x} T_D) \cup \{T_D \text{x}\ P_D)) \to IN_+$ and *continuous set arcs* A_s: $((Pc \text{x} T_D) \cup \{T_D \text{x}\ Pc)) \to IR_+$. The arcs A_d and A_s, are drawn as single arrows. The subset A_h contains the discrete *inhibitory* A_h: $(P_D \text{x} T) \to IN_+$ or continuous *inhibitory* arcs $(P_C \text{x} T) \to IR_+$. These arcs are drawn with a small circle at the end. The subset A_c defines the continuous *flow* arcs A_c: $((P_C \text{ x } T_C) \cup \{T_C \text{ x } P_C)) \to IR_+$, these arcs are drawn as double arrows to suggest a pipe. A *test* A_t input arc is directed from a place of any kind to a transition of any kind, A_t: $(P_D \text{x}\ T) \to IN_+$ or $(P_C \text{x} T) \to IR_+$ and are drawn as dotted single arrows. It does not consume the content of the source place. The arc of net is drawn if the cardinality is not identically zero and non-zero cardinality of arcs is labeled next to the arc with a default value being 1.
- $h: P \cup T \to \{D, C\}$ is a hybrid function, which indicates for each node if P or T is a type discrete, D, or type continuous C. In this case if $p_i \in P_D$ is a discrete place, then the incidence functions $Pre(p_i,t_j)$ and $Post(p_i,t_j)$, $\forall t_j \in T$ are non-negatives integer numbers but if $p_i \in P_C$ is a fluid place, then $\forall t_j \in T_C$ this functions are real positives values; $\forall h(p_i) = D$ and $\forall h(t_j) = C$ is must check the relation $Pre(p_i,t_j)=Post(p_i,t_j)$ is the discrete or continuous *test* arc;
- The complete marking (state) of a net is described by a pair of vectors $M = (\boldsymbol{m}, \boldsymbol{x})$, where the $\boldsymbol{m}$: $T_D \to IN_+$ and $\boldsymbol{x}$: $T_C \to IR_+$ are marking functions of respectively places. The vector–column $\boldsymbol{m} = (m_i, \forall p_i \in P_D)$ is the state listing the contents tokens of discrete places, and vector–column $\boldsymbol{x} = (x_k, \forall p_k \in P_C)$ listing the levels of the fluid places. The initial marking of net is $M_0 = (\boldsymbol{m}_0, \boldsymbol{x}_0)$. The vector $\boldsymbol{m}_0$ gives the initial marking of discrete places. The vector $\boldsymbol{x}_0$ gives the initial marking of fluid places. We denote the set of all *markings* (or the partially discrete and partially continuous state space) of the net by $S = IN_+^{|P_D|} \times IR_+^{|R_C|}$. In the following we de-

note by S_D and S_C the discrete and the continuous component of the state space respectively. The current marking $M = (\boldsymbol{m}, \boldsymbol{x})$ evolves in time. We denote the time by τ, and can think the marking $M(\tau)$ at time τ as the stochastic marking process $S(\tau) = \{ \boldsymbol{m}(\tau), \boldsymbol{x}(\tau), \tau > 0 \}$;

- The function $B : P \to IR_+ \cup \{\infty\}$ describes the fluid lower bound x_i^{min} and upper bounds x_i^{max} on each fluid place. This bound has no effect when it is set to infinity. Each fluid place has an implicit lower bound at level 0;
- $G : T \times S \to \{0, 1\}, S = S_d \cup S_c$, is the *guard function* defined for each transition. For $t \in T$ a guard function $g(t, M)$ is a Boolean function that will be evaluated in each marking, and if it evaluates to *true*, the transition may be enabled, otherwise t is disabled (the default value is *true*).
- *Pri*: $T_D \to IN_+$ defines the priority functions for the firing of each transition. The enabling of a transition with higher priority disables all the lower priority transitions. The set of discrete transitions T_D is partitioned into $T_D = T_\tau \cup T_i$, $T_\tau \cap T_i = \varnothing$ so that: T_τ is a set of timed discrete transitions and T_i *is* a set of immediate discrete transitions. The $Pri(T_i) > Pri(T_\tau)$. A timed transition $t_j \in T_\tau$ is drawn as a black rectangle and has a firing rate is associated to it. An immediate transitions $t_j \in T_i$ is drawn with a black thin bar and has a constant zero firing time
- θ contains parameters defining the firing delay for each timed discrete transition T_τ. It can be marking dependent. Let $T(M)$ denote the set of enabled transitions in $M = (\boldsymbol{m}, \boldsymbol{x})$;
- $W : T_+ \times S_D \to IR_+$ is the weight function of immediate discrete transitions $t_j \in T_i$. If several enabled transitions $t_j \in T_i(M)$ are scheduled to fire at the same time in *vanishing* marking M, with the respective weights w_j, $w(t_k, M) / \sum_{t_j \in T_i(M)} w(t_j, M)$ is the probability to transitions t_k *fire*, there $T(M)$ is the set of enabled transitions in $M = (\boldsymbol{m}, \boldsymbol{x})$.
- $V : T_C \times Bag(P) \to IR_+$ is the marking depedent fluid rate function of timed continuous transitions T_c. This rates appear as labels next to the continuous timed transitions. If $t_i \in T_c$ is enabled in tangible marking M it fires with rate $V_i(M)$.

Figure 1 summarizes the graphical representation of all the *HPN* primitives.

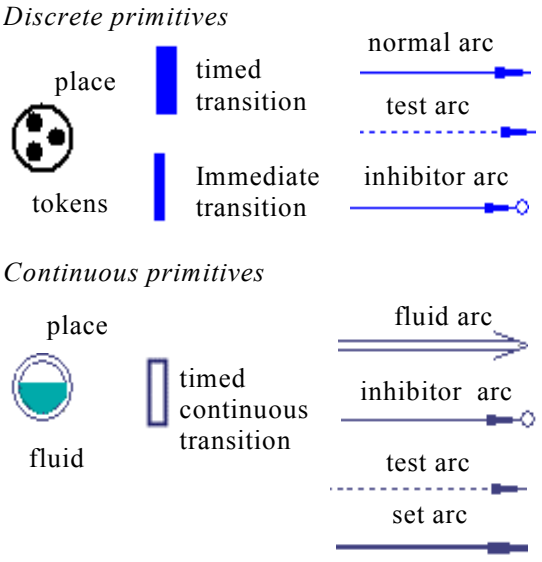

Fig. 1. All the primitive of the *HPN* formalism.

Upon firing, the discrete (respectively continuous) transition removes a specified number (quantity) of tokens (fluid) for each discrete (fluid) input place, and deposits a specified number (quantity) of tokens (fluid) for each discrete (fluid) output place. The levels of fluid places can change the enabling/disabling of continuous transitions;

HPN has five kinds of arcs; *normal discrete* (*continuous set*) *input/output arcs*, *flow* (*continuous*) *input/output arcs*, *discrete* (*continuous*) *inhibitory input arcs* and *discrete* (*continuous*) *test input arcs*. A normal discrete (flow or *continuous set*) input arc (flow) is directed to a discrete (continuous) transition from a discrete place (flow place) from which it consumes respectively a specified number of tokens (quantities of fluid) of the source place by firing of transition. An inhibitory and test input arc is directed from a place of any kind to a transition of

any kind. It does not consume the content of the source place. These five arcs are called *input arcs*. A normal discrete (flow) output arc is directed from a discrete (continuous) transition to a discrete (flow) place which it deposits a specified number of tokens (quantities of fluid) in each respectively output discrete (flow) place by firing of transition.

The role of the previous set and functions will be clarified by providing the enabling and firing rules. Let us denote by m_i the i-th component of the vector $\boldsymbol{m}$, i.e., the number of tokens in discrete place p_i when the marking is $\boldsymbol{m}$, (and x_k denote the k-th component of the vector $\boldsymbol{x}$, i.e., the fluid level in fluid place p_k when the continuous marking is $\boldsymbol{x}$).

Figure 2 summarizes the graphical representation of all the possible ways of placing arcs in a *HPN* net.

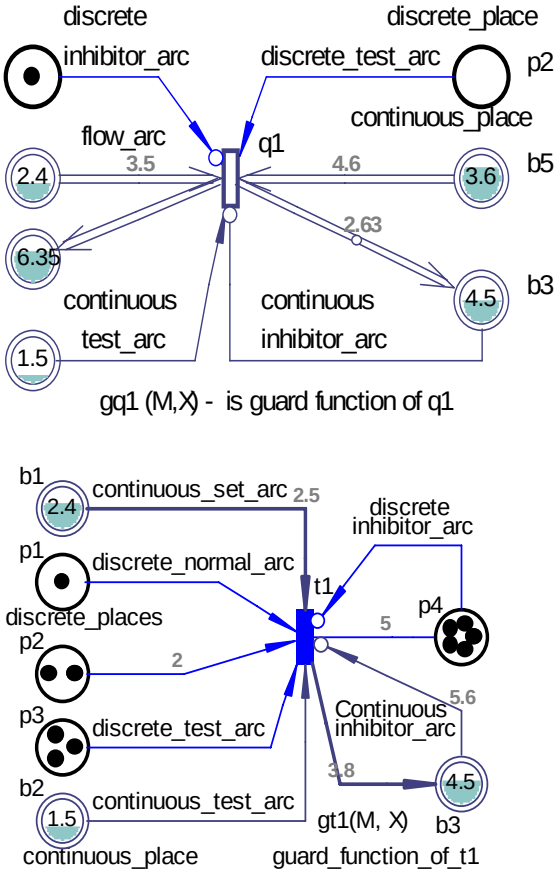

Fig. 2. All the possible ways of placing arcs in a net.

Given a transition $t_j \in T$, we denote with ${}^\bullet t_j = \{p_i \in P_d : A_d(p_i, t_j)>0\} \cup \{p_i \in P: A_t(p_i, t_j)>0\}$ and with $t_j^\bullet = \{p_i \in P_d : A_d(t_j, p_i)>0\}$ the input and output set of transition t_j, and with ${}^\circ t_j = \{p_i \in P : A_h(p_i, t_j)\}$ the inhibition set of transition t_j. The definition of $t_j^\bullet$ involves only discrete places and hence is exactly hence one defined for common *GSPN*. The definitions of ${}^\bullet t_j$ and ${}^\circ t_j$ are instead different since they include also fluid places (see [7]).

We say that a discrete transition $t_j \in T_D$ has *concession* in marking $M = (\boldsymbol{m}, \boldsymbol{x})$ iff:

1) $(\forall p_i \in {}^\bullet t_j, m_i \geq A_d(p_i, t_j)) \,\&$
$(\forall p_i \in {}^* t_j,\ m_i \geq A_t(p_i, t_j)) \,\&\, (\forall p_i \in {}^\circ t_j,$
$m_i < A_h(p_i, t_j)) \& (\forall p_k \in {}^* t_j,)$
$\& (\forall p_k \in {}^\circ t_j, x_k < A_h(p_k, t_j)) \&$
$(\forall p_k \in {}^\# t_j, \forall p_l \in t_j^\#;$
$((x_k \geq A_s(p_k, t_j)) \,\&\, ((B_l - x_l) \geq A_t(t_j, p_l));$

2) $g(t_j,M)=1$, and no transition with higher priority has concession.

If an immediate discrete transition has concession in marking $M = (\boldsymbol{m}, \boldsymbol{x})$, it is enabled the marking is vanishing. Otherwise, the marking is tangible and any timed discrete transition with concession is enabled in it.

We say that a continuous transition $t_n \in T_c$ has *concession* in marking $M = (\boldsymbol{m}, \boldsymbol{x})$ iff:

3) $(\forall p_k \in {}^\bullet t_n, x_k > 0) \,\&\, (\forall p_i \in {}^\circ t_n,$
$m_i < A_h(p_i, t_n)) \& (\forall p_i \in {}^* t_n, m_i \geq A_t(p_i, t_n))$
$\& (\forall p_k \in {}^\circ t_n, x_k < A_h(p_k, t_n)) \& (\forall x_k \in {}^* t_n,$
$x_k \geq A_t(p_k, t_n));$

2) $g(t_n,M)=1$, and no transition with higher priority has concession.

The marking may be changed by firing of enabled transitions. We distinguish two cases.

1. An immediate discrete transition $t_j \in T_i$ enabled in marking $M = (\boldsymbol{m}, \boldsymbol{x})$ yields a new marking $M' = (\boldsymbol{m'}, \boldsymbol{x})$, i.e., the firing of an immediate discrete transition does not change the continuous part of the marking. We can write $(\boldsymbol{m}, \boldsymbol{x})$ $[t_j > (\boldsymbol{m'}, \boldsymbol{x})$.

2. If marking $M = (\boldsymbol{m}, \boldsymbol{x})$ is tangible, fluid could continuously flow through the flow arcs A_c of enabled continuous transitions into or out of

fluid places. As a consequence of this, a transition t_c is *enabled* at M *degree* iff for every $p_c \in {}^{\bullet}t_c$, $\boldsymbol{x}(p_c)>0$, *and its enabling degree* is enab(t_c ,M)=$\min_{p_c \in {}^{\bullet}t}\{\boldsymbol{x}(p_c)/Pre(p_c, t_c)$. In the *HPN*, the potential rate $\beta_i(M)$ of change of fluid level in place $p_i \in P_C$ in marking M are deterministic and is given by $\beta_i(M) = \sum_{t_k \in T(M)} [V_{k,i}(M) - V_{i,k}(M)]$, where , for any given $t_k \in T_C$, $V_{k,i}(M)$ is an input fluid rate of fluid place $p_i \in P_C$ and $V_{i,k}(M)$ is an output fluid rate of this place. We allow the firing rates and the enabling functions of the timed discrete transitions, the firing speeds and enabling functions of the timed continuous transitions, and arc cardinalities to be dependent on the current state of the *HPN*, as defined by the marking $M(\tau)$.

In the following, we describe the *HPN* visual simulation using a software Visual Petri Nets+ (VPNP) tool [5].

3. VPNP tool overview

VPNP is a window-based, object-oriented tool, in which elements typical of hybrid Petri net models (places, transitions, and arcs) are manipulated under the assistance of basic syntactical rules that prevent the construction of incorrect models. This tool is designed with a broad functionality: it contains a large number of various extensions to stochastic and timed hybrid Petri nets that have been proposed, user-defined place and transition rewards, place capacities, discrete and continuous arc marking-dependent multiplicities, firing marking-dependent firing discrete and flows rates, single and infinite-server transitions, and memory policies.

The model drawn using the interface is internally translated into several files that are however completely transparent to the user.

The representation of *HPN* model is thus the object to which all the command provided by the Graphical User Interface (*GUI*). The *GUI* is shown in Figure 3.

Heaving specified the *HPN* model, it can be saved into a data-base from which it can be retrieved for any subsequent use.

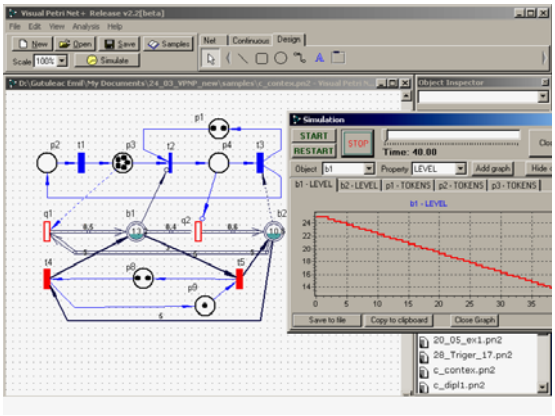

Fig. 3. Graphical User Interface of VPNP.

The simulation of a *HPN* model is performed in *VPNP* by first constructing the model on the screen of a graphical workstation using the facilities provided by the graphical user interface. Once that the simulation model has been specified, a debugging phase must take place to insure that it is first syntactically, and then semantically correct. Finally, the actual simulation is performed of the model are collected and analyzed to obtain estimates for the performance indexed that are relevant for the study.

Constructing *HPN* models consists of drawing places and transitions connected by arcs. This is made possible by *VPNP* that provides a *drawing board* where graph nodes of *HPN* selected from a menu of basic objects can be deposited. To facilitate the drawing of graphically well organized models, a grid can be superimposed on its corners. Discrete direct (fluid and continuous set) arcs that can be of input, output, and discrete (continuous) test and inhibition type connect nodes. They are drawn selecting first the origin node, and subsequently the destination node. Arcs behave as rubber-bands and can look like segmented lines though the use of intermediate points. Objects deposited on the drawing board can be repositioned by dragging them around with usual point-and-select methods. Arcs stick to the origin and destination nodes so that moving nodes affects connecting arcs as well. To make the model look nicer, segmented arcs can be turned into continuous (curved) lines by means of a *spline* option that smooths their angles. Group of elements (subnets) can be graphically manipulated as a single object using

a multiple selection option which then allows to move all the elements of a group. These operations on global objects preserve the relative positions of the component elements. Selected groups of elements can be removed from the drawing with a single operation.

An important aspect of timed *HPN* models is the specification of the initial making of the net that is relevant both as the starting point for the future simulation experiment, and as the specification that provides additional meaning to certain elements of the net and thus identifies the set of possible states of the model. The initial marking is obtained editing a lace element parameter to specify the number of tokens or fluid level that is initially contained in each type place. A default value of zero tokens is introduced to make the specification of the initial marking quicker. The specification of the model is completed with the definition of the parameters and of the performance indexes relevant for the study. All the transitions of the net possess default parameter values that can be modified with simple editing actions. When indications exist that the model will be used for several studies in which the effect of the variation of certain parameters will be investigated, these parameters can be specified as variables, using the *VPNP* object inspector.

4. Visual simulation of timed HPN

The specification of a model using a *HPN* based formalism has the advantage of an explicit representation of the relations among the different models of the net (interconnections) and of the activities that are currently taking place (tokens and/or fluid). Moreover the dynamics of the model is completely represented by the movement of such tokens. Advanced graphics allows exploit these advantages recognizing their importance in the description of a model and in the understanding of its behaviour.

Visual simulation is a method that exploits the graphical description of the model to show the dynamic evolution that takes place during its simulation directly on the graphical representation. Static graphical facilities are used to describe the topology of the *HPN* model. The dynamic graphical facilities are used to visualize the movement of the tokens (fluid), to provide snapshots of the model, and to represent the evaluation of computed performance. Visual simulation is very useful during the early stage of the construction of a model since it represents a powerful tool for its debugging. Moreover, the animation of a correct simulation model may provide important insights on the behavior of the actual system, especially because of the possibility of highlighting particular evolution patterns that may be crucial for the good (or bad) behavior of the system, but difficult to deduce from average performance indexes obtained with standard methods. . The simulation is a stochastic experiment in which is performed to achieve necessary data. Also in this mode the user can see consecutive change of markings in the nets, exalted transitions and other aspects of Petri net principles, which makes this analysis mode more educative and attractive for those new to Petri nets formalism.

VPNP at the moment is still in the development stage and some its modules, or even most of its structure can undergo considerable changes. But the idea of building a user-friendly, handy tool for analyzing stochastic and timed *HPN*. Algorithms of functioning of the editor of a *HPN*s and the inspector of objects are tool complex and insignificant in a theme of the given degree work and consequently here to not be considered. Let's consider algorithm of functioning of a simulator of a network. Feigning represents the process of a net in real time, also in a mode of accelerated or slowed down time. Feigning begins with initial marks (which it is restored after end of process). By means of process of simulation it is possible evidently representing the process of transition of a timed *HPN* of one condition in another, also to check correctness of functioning and to collect the various information in real-time. As some transitions can be in the conflict before the operation, of each following transition there is a check on his opening.

The computation of the marking-dependent flow rate across a continuous arc (the continuous throughput), requires a more detailed analyses. A problem arises when the fluid level reaches one of the boundary conditions. Consider the input fluid places $b_i \in {}^\bullet u_j$ and the output fluid

places $b_k \in u_j^{\bullet}$ of a continuous transition $u_j \in T_C$, and the actual fluid levels of this places are respectively $x_i^{min} < x_i < x_i^{max}$ and $x_k^{min} < x_k < x_k^{max}$.
We require that for every transition $u_j \in T_C$:
$\delta_j = \min_{\forall b_i} \{ (x_i - x_i^{min}) / Pre(b_i, u_j) \}$ and
$\varepsilon_j = \min_{\forall b_k} \{ (x_k - x_k^{max}) / Post(u_j, b_k) \}$,
$\eta_j = \min_{\forall b_k} \{ \delta_j / \varepsilon_j \}$. The actual fluid rate V_j^* of a continuous transition $u_j \in T_C$ is determinate in the following: *if* $(\eta_j < V_j)$ *then* $V_j^* = \eta_j$ *else* $V_j^* = V_j$, where V_j is potential flow rate function of this continuous transitions.
In the case of structural and effective conflicts, $|b_i^{\bullet}| > 1$ these are solved using the *priority semantic* or *proportional semantic* [1]. For example we consider the fluid place $b_i \in P_C$ with fluid level $x_i^{min} < x_i < x_i^{max}$, multiple inputs $\{u_2, u_4\} \in {}^{\bullet}b_i$, and outputs $\{u_1, u_3\} \in b_i^{\bullet}$ continuous transitions with the potential flow rate V_j.
Let $\alpha_j = V_j \cdot Post(u_j, b_i)$ and $\beta_k = V_k \cdot Pre(u_k, b_i)$, i = 2,4 and k =1,3. If $\alpha_2 + \alpha_4 < \beta_1 + \beta_3$, the fluid level as function of the time decreases with rate $(\alpha_2 + \alpha_4) - (\beta_1 + \beta_3)$, until it reaches the boundary value $x_i = x_i^{min}$. From this point on, the potential flow rate is set to 0 and place b_i remains constantly empty. In order to compute the flow throughput along the arcs, the individual flow rates along each arc must be redefined in such a way that the difference between input and output rates remains equal to 0. By renaming the new output flow rates $\beta_1^* + \beta_3^*$, different semantics can be adopted [1] to redistribute the flow rate such the following relation is verified: $(\alpha_2 + \alpha_4) = (\beta_1^* + \beta_3^*)$. Under a priority semantic, the input flows are used to saturate the output flows in a predefined order. If for example the transition u_1 has priority over the u_3, $Pri(u_1) > Pri(u_3)$, then: if $(\alpha_2 + \alpha_4) \le \beta_1$, then $\beta_1^* = \alpha_2 + \alpha_4$ and $\beta_3^* = 0$; if $\beta_1 < \alpha_2 + \alpha_4 \le \beta_1 + \beta_3$ then $\beta_1^* = \beta_1$ and $\beta_3^* = \alpha_2 + \alpha_4 - \beta_1$.
Under a proportional semantic instead, the input flow is proportioned in away that keeps constant the ratio between the output flows, that is:
$\beta_k^* = \beta_k \cdot (\alpha_2 + \alpha_4) / (\beta_1 + \beta_3)$, k = 1, 3.
Simulation is started from the separate form of a Figure 4.

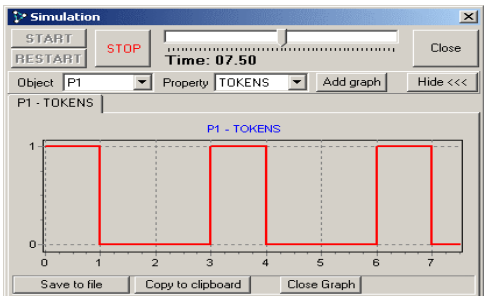

Fig. 4. The form of simulation.

The description of the form: START - start of a network on simulation; RESTART - start of a network on simulation with restoration of initial marking; STOP - stops of simulation, can be continued by button START; *Time* - modeling time; the *Toddler* - sets speed of simulation; Close - to close a Window. The panel for creation of diagrams as a chosen parameter is displayed in the diagram. For a choice of parameter all over again it is necessary to choose an object, then his parameter and press the button *Add graph*. For displaying a part of the form with diagrams press " *Show* > > > ". If the diagrams are not necessary at present it is possible to hide them, pressing " *Hide* < < < " to a Figure 4. If the diagram is used for inserting into a report or in another program, it can be done with the help of the buttons "*Save to file*" - preservation of the diagram in a file of a format *name.bmp* or copy to clipboard - to copy in the buffer of an exchange. The Simulator supports all types of objects realized in the program.
The step of time simulation gets out automatically, on the following rule: The *minimal step*: = *MAXINT*; For all transitions the beginning: If transition time that *temp: = 1/Rate*; If transition continuous that *temps*: = 1; If *temp* < *Minimal step* then *Minimal step*: = *temp*; The *end.*
However, it may occurrence the situations when only immediate transitions are enabled. Then time costs on a place and if in this case there is even one enabled timed transition, it will not be fired. When a discrete or continuous transition

fires, tokens or fluid will be symbolically moved along the input arcs from the input places to the transition and along the output arcs from the transition to the output places, according to the various arc multiplicities. Viewing the simulation visualization in this manner significantly enhances the user's understanding of the nets dynamic behavior, and is very useful in presentations to non-specialists and of course for functional verifying (debugging) the model behavior.

5. Conclusions

In this paper we have discussed the possibility of timed *HPN* models to study the performance characteristics of hybrid systems. When dealing with real cases, it often happens that the *HPN* models tend to become very complex. In these cases *HPN* models can be used as formal specification of system and automatically translated into detailed simulation programs in order to estimate performance result.

The construction and the analysis of complex timed *HPN* models can only be done with the help of powerful automatic tools such as the user-friendly software *VPNP* tool whose performance modeling facilities and software architecture have been illustrated in this paper.

The integration within the same interface of the graphical facilities for the model construction, of the structural analysis algorithms for model validation, and of the control panels for performing the visual and interactive simulation, emphasizes the importance of including in a simulation experiment both validation and evaluation aspects of timed *HPN* models .

The animation of the model, performed only when desired by the user, appears to be a powerful tool that complements the structural results for the debugging and tuning of the models used for the performance analysis.

Already *VPNP* is used in educational process; it gives students a grasp of hybrid Petri net principles, introducing them into world for performance evaluation of computer systems and communication networks.

References

[1] Alla, A., David, H. (1998) *Continuous and hybrid Petri nets*. Journal of systems Circuits and Computers, 8(1), pp. 159-188.

[2] Ajmone Marsan M., Balbo G., Conte G and Donatelli S. (1992) *Modelling with Generalized Stochastic Petri Nets.* Series in Parallel Computing. John Wiley & Sons.

[3] David, R. (1997) *Modeling Hybrid Systems using Continuous and Hybrid Petri Nets* In:Proc. PNPM', pp.135-144.

[4] Drath, R. (1998) *Hybrid object nets: An object oriented concept for modeling complex hybrid systems*. In.: Proc. Hybrid Dynamical Systems, 3rd International Conference on Automation of Mixed Processes, pp. 437-442.

[5] Guţuleac, E., Reilean, and A. Boşhneaga C. (2002) *VPNP - software tool for modeling and performance evaluation using generalized stochastic Petri nets*. In: Proc. of 6th International Conference on DAS2002, Suceava, România, pp. 243-248.

[6] Gutuleac E. (2000) *Analyse de performances d`un système multiprocesseur par réseaux de Petri hybrides stochastiques*. In: Proc. of 5th DAS2000, Suceava, România, pp. 292-297.

[7] Gribaudo, M., Sereno, M., Horvath, A., and Bobio, A. (2001) *Fluid stochastic Petri nets augmented with flush-arcs: Modelling and analysis*. Discrete Event Dynamic Systems, vol. 11, no.1/2, pp. 97-117.

[8] Sanders W. S., Obal II W. D., Qureshi M. A. and Widjanarko F. W. (1995). *The UltraSAN Modeling Environment.* Performance Evaluation, 24(1), pp. 89-115.

[9] Wolter K., Hommel G. (1997) *Hybrid Modelling with Second Order Fluid Stochastic Petri Nets.* In Proc. Workshop on Parallel and Distr. Real-Time Systems, pages. 239-243,

[10] Zimmermann A. and Freiheit J. (1998) *TimeNETms An integrated modelling and performance evaluation tool for manufacturing systems*, in IEEE Int. Conf. On Systems, Man, and Cybernetics, San Diego, USA, pp. 535-540.