Chapter 16

Timed Petri Nets

José Reinaldo Silva and Pedro M. G. del Foyo

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50117

1. Introduction

In the early 60's a young researcher in Darmstadt looked for a good representation for communicating systems processes that were mathematically sound and had, at the same time, a visual intuitive flavor. This event marked the beginning of a schematic approach that become very important to the modeling of distributed systems in several and distinct areas of knowledge, from Engineering to biologic systems. Carl Adam Petri presented in 1962 his PHD which included the first definition of what is called today a Petri Net. Since its creation Petri Nets evolved from a sound representation to discrete dynamic systems into a general schemata, capable to represent knowledge about processes and (discrete and distributed) systems according to their internal relations and not to their work domain. Among other advantages, that feature opens the possibility to reuse some experiences acquired in the design of known and well tested systems while treating new challenges.

In the conventional approach, the key issue for modeling is the partial ordering among constituent events and the properties that arise from the arrangement of state and transitions once some basic interpretation rules are preserved. Such representation can respond from several systems of practical use where the foundation for analysis is based in reachability and other property analysis. However, there are some cases where such approach is not enough to represent processes completely, for instance, when the assumption that all transitions can fire instantaneously is no longer a good approximation. In such cases a time delay can be associated to firing transitions. This is absolutely equivalent (in a broader sense) to say that firing pre-conditions must hold for a time delay before the firing is completed. The first approach is called T-time Petri Net and the second P-time Petri Nets.

Thus, what we have in conclusion is that even in a hypothesis that we should consider only firing pre-conditions[1] [31][19] a time delay is associated with a transition location and consequently to its firing. Several applications in manufacturing, business, workflow and

[1] In many text books and review articles the enabling condition is presented using only firing pre-conditions as a requirement. This can be justified since the use of this week firing condition is sufficient if a complete net, that is, that includes its dual part, is used

other processes can use this approach to represent processes in a more realistic way. It is also true tat even with a simple approach a strong representation power can be derived, including the possibility to make some direct performance analysis [29][42]. This is called a time slice or a time interval approach. In general, this augmented nets with time delay (P-time, T-time or even both) are called Timed Petri Nets[2].

There are also cases where it is necessary to use more than time delays. In such cases the time is among the variables that describes the state (a set of places in Petri Nets). Notice that raising the number of variables that characterize a state would make untreatable the enumeration of a net state space. Therefore, a more direct approach is adopted, where each transition is associated to a time interval t_{min} and t_{max} where the first would stand for the minimal waiting time since the enabling until a firing can occur. Similarly, t_{max} stands for the maximum waiting time allowed since enabling up to a firing.

If the time used in the model is a real number, then we call that a Time Petri Net. It should be also noticed that if $t_{min} = t_{max}$ the situation is reduced to the previous one where a deterministic time interval is associated to a transition. Thus, Time Petri Net is the more general model which can be used to model real time systems in several work domains, from electronic and mechatronic systems to logistic a business domains.

In this chapter we focus in the timed systems and its application which are briefly described in section 2. In section 3 we will present a perspective and demand for a framework to model, analysis and simulation of timed (and time) systems mentioning the open discussion about algorithms and approaches to represent the state space. That discussion will be oriented by the recent advances to establish a standard to Petri Nets in general that includes Timed and Time Petri Nets as a extension. Such standard is presented in ISO/IEC 15.909 proposal launched for the first time in 2004. A short presentation of what could be a general formalization to Time Petri Nets is done in section 4. Concluding remarks are in section 5.

2. A schematic description of time dependent systems

Petri Nets are an abstract formal model for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic [31]. Since its creation the formalism has been extended by practitioners and theoreticians for dealing with complex systems attached to many application fields. One of those important extensions were proposed to deal with timed systems.

Among several proposed extensions to deal with time we detach two basic models: Ranchamdani's *Timed Petri nets* [34] and Merlin *Time Petri nets* [30]. These two temporal Petri net models are included in t-time nets because time inscriptions are always associated to transitions. Other time extensions have been published including some approaches where time is associated to places or even to both places and arcs (see [13] for a survey).

Formally, Petri Nets can defined as:

[2] Some authors also include another possibility where time is associated to the arcs, that is, to a pair (x, y) where $x, y \in X = S \subseteq T$ where S and T denotes the set of places and transitions, respectively.

Definition 1. [Petri Net] A Petri net structure is a directed weighted bipartite graph

$$N = (P, T, A, w)$$

where

P is the finite set of places, $P \neq \emptyset$

T is the finite set of transitions, $T \neq \emptyset$

$A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs from places to transitions and from transitions to places

$w : A \to \{1, 2, 3, \ldots\}$ is the weight function on the arcs.

We will normally represent the set of places by $P = \{p_1, p_2, \ldots, p_n\}$ and the set of transitions $T = \{t_1, t_2, \ldots, t_m\}$ where $|P| = n$ and $|T| = m$ are the cardinality of the respective sets. A typical arc is of the form (p_i, t_j) or (t_j, p_i) according to arc direction, where its weight w is a positive integer greater than zero.

Definition 2. [Marked Petri Net] A marked Petri net is a five-tuple (P, T, A, w, M) where (P, T, A, w) is a Petri Net and M is a marking, defined as a mapping $M : P \to \mathbb{N}^+$

Thus, a marking is a row vector with $|P|$ elements. Figure 1 shows a possible marking for a simple Petri Net.

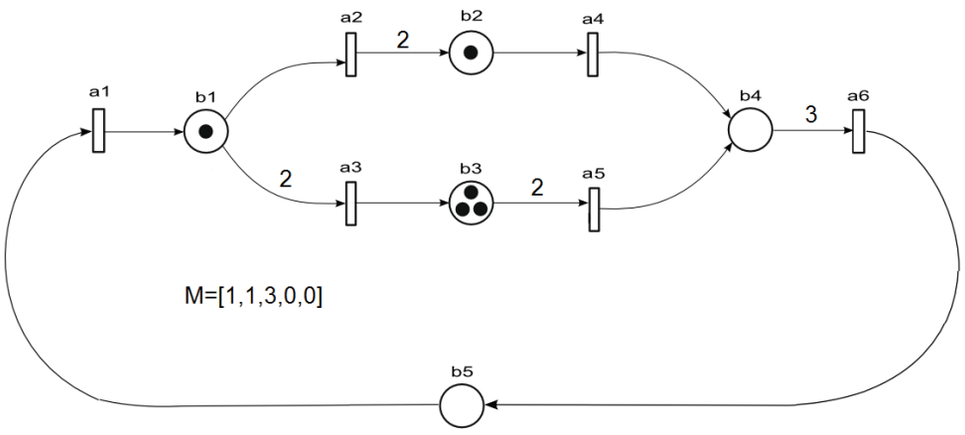

Figure 1. A marked Petri net and it respective marking vector M

The relational functions $Pre, Pos : P \times T \to \mathbb{N}$ are defined to obtain the number of tokens in places p_i, p_j which are preconditions or postconditions of a transition $t \in T$, that is, there exists arcs $(p_i, t), (p_j, t) \in A$ for the Pre function or $(t, p_i), (t, p_j) \in A$ for the Pos function. In Fig. 1 for instance we have $Pre(b1, a3) = 1$ and $Pos(b3, a3) = 3$.

Using Petri nets to model systems imply in associate net elements (places or transitions) to some components and actions of the modeled system, turning out in what is called "labeled" or "interpreted" nets. The evolution of marking in a labeled Petri nets describes the dynamic behavior of the modeled system.

We restrict the definition of Labeled Petri Net to associate labels only to events or actions similarly to the formalism of automata.

Definition 3. [Labeled Petri Net] A labeled Petri net is a seven-tuple

$$N = (P, T, A, w, E, l, M_0)$$

where

(P, T, A, w) is a Petri net structure
$E \subseteq \mathbb{P}(T)$, $E \neq \emptyset$
$l : T \to E$ is the transition labeling function
$M_0 : P \to \mathbb{N}^+$ is the initial state of the net

Labeled Petri Nets has been proved to be an efficient tool for the modeling, analysis and control of Discrete Event System (DES). Petri Nets is a good option to model these DES systems for wide set of applications, from manufacturing, traffic, batch chemical processes, to computer, communications, database and software systems [27]. From now on we shall refer to "labeled Petri nets" simply as "Petri nets".

The state transition mechanism in Petri nets is provided by moving tokens through the net and hence changing to a new state. When a transition is enabled, we say that it can fire or that it can occur.

Definition 4. [Enabled Transition] A transition $t_j \in T$ in a Petri net is said to be enabled if

$$\forall p \in P, \quad M(p) \geq Pre(p, t_j)$$

In other words, a transition t_j in the Petri net is enabled when the number of tokens in p is greater then or equal to the weight of the arc connecting p to t_j , for all places p that are input to transition t_j.

The set of all enabled transition at some marking M is defined as $enb(M)$. In Fig. 1 only transitions $a2$, $a4$ and $a5$ are enabled, then $enb(M) = \{a2, a4, a5\}$.

Definition 5. [State Transition] A Petri net evolves from a marking M to a marking M' through the firing of a transition $t_f \in T$ only if $t_f \in enb(M)$. The new marking M' can be obtained by

$$\forall p \in P | (p, t_f) \vee (t_f, p) \in A, M'(p) = M(p) - Pre(p, t_f) + Pos(p, t_f)$$

The reachable markings in a Petri net can be computed using an algebraic equation. Two incidence matrices must be defined (A^- for incoming arcs, and A^+ for outgoing arcs) and a firing vector u which is a (unimodular) row vector containing "1" in the corresponding position of the firing transition and "0" in all other positions.

The new marking can be obtained using the state equation:

$$M' = M + u(A^+ - A^-) \tag{1}$$

This formalism is sufficient to represent a great amount of dynamic discrete systems, based only in partial order sequence of transitions. However, "untimed"[3] Petri nets are not powerful enough to deal with performance evaluations, safety determination, or behavioral properties in systems where time appears as a quantifiable and continuous parameter.

Ramchandani's timed Petri nets were derived from Petri nets by associating a firing finite duration to each transition in the net. Timed Petri nets and related equivalent models have been used mainly to performance evaluation [7].

Definition 6. [Timed Petri Net] A timed Petri net is a six-tuple

$$N = (P, T, A, w, M_0, f)$$

where

(P, T, A, w, M_0) is a marked Petri net

$f : T \to \mathbb{R}^+$ is a firing time function that assigns a positive real number to each transition on the net

Therefore, the firing rule has to be modified in order to consider time elapses in the transition firing. If an enabled transition $t_j \in enb(M)$ then it will fire after $f(t_j)$ times units since it became enabled. The system state is not only determined by the net marking but also by a timer attached to every enabled transition in the net.

Definition 7. [Clock State] The clock state is a pair (M, V), where M is a marking and V is a clock valuation function, $V : enb(M) \to \mathbb{R}^+$

For a clock state (M, V) and $t \in enb(M)$, $V(t)$ is the value of the clock associated with a transition t. The initial clock state is $s_0 = (M_0, V_0)$ where $V_0(t) = f(t), \forall t \in enb(M_0)$.

Definition 8. [New Enabled Transition] A transition $t \in T$ is said new enabled, after firing transition t_f at marking M which leads to marking M', if it is enabled at marking M' and it was not enabled at M or, if it was enabled at M, it is the former fired transition t_f. Formally:

$$new(M') = \{t \in enb(M') | t = t_f \vee \exists p,\ (M'(p) - Pos(p, t_f)) \leq Pre(p, t)\}$$

We denote as $new(M')$ the set of transitions new enabled at marking M'.

The reachability graph of a timed Pedri net can be computed using the definition of firable transition, that is, those transitions that can be fired in a certain marking.

Definition 9. [Firable Transition] A transition $t_f \in T$ can fire in a marking M yielding a marking M' if:

$t_f \in enb(M)$

$V_M(t_f) \leq V_M(t_i)\ \forall t_i \in enb(M)$

[3] A Petri Net where there is no event depending directly or parametrically of the time

We denote as $\Upsilon(M)$ the set of transitions firable at a marking M. Assuming that the firing of transition t_f leads to a new marking M', we denote it as $(M, V_M) \xrightarrow{\tau} (M', V_{M'})$ where $\tau = V_M(t_f)$ is the time elapsed in state transition, M' is computed using equation 1 and $V_{M'}$ is computed as follows:

$$V_{M'} = \begin{cases} f(t) & \text{se } t \in new(M'); \\ V_M(t) - \tau & \text{se } t \in enb(M') \setminus new(M') \end{cases}$$

The reachability tree can be built including all feasible firing sequences of the timed Petri net. Notice that in M_0 all transition are new enabled, $enb(M_0) = new(M_0)$. However, finite reachability trees can be built only for bounded Petri nets (see bounded property in [31], and an equivalent result for Time Petri in [6]).

Paths or runs in a reachability tree are sequences of state transitions in a timed Petri net. Then, the time elapsed in some path can be computed as the summation of all the elapsed times in the firing schedule. If a path $\omega = s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \xrightarrow{\tau_3} s_3$ exists, then the time elapsed between states s_0 and s_3 is $\tau = \tau_1 + \tau_2 + \tau_3$.

The reachabitity tree approach has been successfully used in communication protocols validation and in performance analyses [38, 45]. Moreover, these tests require known computation times for the tasks (often referred by WCET as "Worst Case Execution Time") or process durations. Besides the difficulty to measure or estimate such times, taking into consideration a deterministic time (even in the longest path) does not lead necessarily to the worst case [36].

More realistic analysis can be done on communication protocols using Merlin approach, since some network time durations or even software routines cannot be completed always in the same time [6, 30].

Merlin defined Time Petri Nets (TPN) as nets with a time interval associated to each transition. Assuming that a time interval $[a, b]$ $(a, b \in \mathbb{R}^+)$ is associated with a transition t_i, and that such transition has been enabled at a marking M_i and is being continuously enabled since then in all successive markings $M_{i+1}, \ldots, M_{i+k}$, we define:

- a $(0 \leq a)$, as the minimal time that transition t_i must remain continuously enabled, until it can fire. This time is also known as Early Firing Time (EFT)
- b $(0 \leq b \leq \infty)$, as the maximum time that transition can remain continuously enabled without fire. This time is also known as Latest Firing Time (LFT)

Times a and b for transition t_i are relative to the moment in which transition t_i became last enabled. If transition t_i became enabled at time τ, and remains continuously enabled at $\tau + a$ then it can be fired. After time $\tau + a$, transition t_i can remains continuously enabled without fire until $\tau + b$, in which it must be fired. Note that transitions with time intervals $[0, \infty]$ correspond to the classical "untimed" (no deterministic) Petri net behavior.

The firing semantic described here is called "strong semantic". There also exists a called "weak semantic" in which transitions must not necessarily be fired at its LFT, and after that time it can no longer be fired [35]. In this chapter we will used the strong semantic.

Time Petri nets then can model systems in which events has non-deterministic durations. State transitions in that kind of systems may occur not in an exact time but in some time interval. Real-time systems are examples of this kind of system.

Definition 10. [Time Petri Net] A time Petri net is a six-tuple

$$N = (P, T, A, w, M_0, I)$$

where

(P, T, A, w, M_0) is a marked Petri net

$I : T \to \{\mathbb{R}^+, \mathbb{R}^+ \cup \{\infty\}\}$ associates with each transition t an interval $[\downarrow I(t), \uparrow I(t)]$ called its static firing interval. The bounds of the time interval are also known as EFT and LFT respectively.

The enabling condition remains the same as in the timed Petri Net but the firing rule must be redefined. The possibility to fire in a time interval rather than an exact time lead to the existence of infinite clock states. Then, even for bounded Petri nets the state space will be infinite, turning intractable any analysis technique based on that model formalism. To overcome this problem, Berthomieu and Menasche [7] proposed a new definition for state.

Definition 11. [Interval State] A state in a TPN is a pair (M, θ) where

M is a marking

θ is a firing interval function $\theta : enb(M) \to \{\mathbb{R}^+, \mathbb{R}^+ \cup \{\infty\}\}$.

The firing interval associated with transition $t \in enb(M)$ is $\theta(t) = [\downarrow \theta(t), \uparrow \theta(t)]$.

Using that approach, a bounded TPN yields a finite number of states [6]. Note that each Interval state contains infinite clock states, then the new state definition allow us to group infinite clock states into one interval state satisfying the condition:

$$(M, V) \in (M', \theta) \text{ iff } (M = M') \wedge \forall t \in enb(M), \downarrow \theta(t) \leq f(t) - V(t) \leq \uparrow \theta(t)$$

An enumerative analysis technique was introduced in [7] based in what is called "state classes". An algorithm for enumeration of these state classes was proposed for bounded TPNs and then used to the analysis of system. Since then, many algorithm has been proposed to build system state space based on the state class approach [6, 9, 11, 17, 22, 44].

Definition 12. [Firable Transition] Assuming that transition $t_f \in T$ becomes enabled at time τ in state (M, θ), it is firable at time $\tau + \lambda$ iff:

$t_f \in enb(M)$: t_f is enabled at (M, θ).

$\forall t_i \in enb(M), \downarrow t_f \leq \lambda \leq min(\uparrow t_i)$

We denote as $\Upsilon(s)$ the set of transitions firable at state $s = (M, \theta)$ and as $(M, \theta) \xrightarrow{t_f, \lambda} (M', \theta')$ the behavior "transition t_f is firable from state (M, θ) at time λ and its firing leads to state (M', θ')"

The first condition is the usual one for Petri nets and the second results from the necessity of firing transitions according to their firing interval. According to the second condition, a transition t_f, enabled by a marking M at absolute time τ, could be fired at the firing time λ iff λ is not smaller than the EFT of t_f and not greater than the smallest of the LFT's of all the transitions enabled by marking M.

Each firable transition will have its own time interval in which it can be fired. That time depends of its EFT and of the time elapsed since it became last enabled, and of the time in which the rest of the the enabled transitions will reach its LFTs, also according to the time elapsed since each one became last enabled.

Definition 13. [State Class] A state class is a pair $C = (M, D)$ where:

M is a marking;

D is the firing domain of the class.

The state class marking is shared by all states in the class and the firing domain is defined as the union of the firing domains of all the states in the class. The domain D is a conjunction of atomic constraints of the form $(t - t' \prec c)$, $(t \prec c)$ or $(-t \prec c)$, where $c \in \mathbb{R} \cup \{\infty, -\infty\}$, $\prec \in \{=, \leq, \geq\}$ and $t, t' \in T$.

The domain of D is therefore convex and has a unique canonical form defined by:

$$\bigwedge_{(t,t') \in enb(M)^2} t - t' \leq Sup_D(t - t') \wedge \bigwedge_{(t \in enb(M))} t \leq Sup_D(t) \wedge -t \leq Sup_D(-t)$$

where $Sup_D(t - t')$, $Sup_D(t)$, and $Sup_D(-t)$ are respectively the supremum of $t - t'$, t, and $-t$ in the domain of D.

In [12] was proposed an implementation for the firing rule, which directly computes the canonical form of each reachable state class in $O(n^2)$. The firing sequences beginning at some state s_i has the form:

$$\omega = s_i \xrightarrow{t_1, [l_{i+1}, u_{i+1}]} s_{i+1} \xrightarrow{t_2, [l_{i+2}, u_{i+2}]} s_{i+2} \dots s_{i+n-1} \xrightarrow{t_j, [l_{i+n}, u_{i+n}]} s_{i+n} \quad (2)$$

where the intervals $[l_n, u_n]$ for each t_n are respectively the minimum and maximum time in which the transition can fire.

Once the state space is built, different verifications can be done, including model-checking techniques which determine if some temporal formulas are true or false over the state space. There are severals tools that use such approach [8, 17, 20, 21, 44].

In the special case where EFT and LFT has the same value, the behavior of the TPN reproduce the one of timed Petri Nets. Note that in paths over the graph which is built using the timed Petri net firing rule, the elapsed time in state transitions is fixed while in the TPN it is a time interval. In timed Petri nets states are clock states while in TPN they are interval states or state classes that contain infinite clock states.

The Timed Automaton (TA) with guards [3] is an automaton to which is adjoined a set of continuous variables whose dynamical evolution is time-driven. This formalism has been

used to modeling and to a formal verification of real-time systems with success. Some tools as KRONOS [16] and UPPAAL [28] are available for such purposes. The state space yielded using Timed Automaton with guards is quite similar of that of TPN regarding the differences on their constructions.

3. Towards a unified PN system framework

In spite of the great theoretical importance and applicability of Time (or Timed) Petri Nets the PN theory was develop since the early 60's in different directions, always seeking for a way to face combinatorial explosion or to approximate to Fuzzy Logic or object-oriented systems. Several extensions were developed to fit practical applications or to attend the need to treat a new class of distributed systems, such as real-time systems. The new century started with a good amount of work published in this area but also with some confusion about concepts and representations. On the other hand, the raising complexity of distributed systems demanded a unified approach that could handle from abstract models down to the split of these general schemas in programs addressed to specific devices. In fact, integrated and flexible systems depend on that capacity.

A ISO/IEC project were launched in the beginning of this century to provide a standard to Petri Nets: the ISO/IEC 15909. Briefly, this project consists of three phases, where the first one defined P/T nets and High Level Nets in a complementary view, that is, taken P/T nets as a reduced set of the High Level Nets (HLPNs) when we reduce the color set to only one type. That is equivalent to unfold the net. Therefore, the proposed standard provides a comprehensive documentation of the terminology, the semantical model and graphic notations for High-level Petri nets. It also describes different conformance levels. Technically, the part 1 of the standard provides mathematical definitions of High-level Petri Nets, called semantic model, and a graphical form, known as High-level Petri Net Graphs (HLPNGs), as well as its mapping to the semantic model [23, 24].

Similarly to other situations where advances in technology and engineering demands a standardization, the introduction of a Petri Net standard also put in check the capacity of exchanging models among different modeling environments and tools. Thus, a Petri Net Markup Language (PNML) was introduced as an interchange format for the Petri nets defined in part 1 of the standard [25]. That composes the Part 2 of the standard and was published in February 2011, after a great amount of discussion, defining a transfer format to support the exchange of High-level Petri Nets among different tool environments [24]. The standard defined also a transfer syntax for High-level Petri Net Graphs and its subclasses defined in the first part of the standard, capturing the essence of all kinds of colored, high-level and classic Petri nets.

Part 3 is of the standard is devoted to Petri nets extensions, including hierarchies, time and stochastic nets, and is still being discussed, with a estimated time to be launched in 2013. The main requirement is that extensions be built upon developments over the core model, providing a structured and sound description. That also would allow user defined extensions based on built-in extensions and would reduce the profusion of nets attached to application domains. At least two main advantages would come out from that:

- a simple, comprehensive and structured definition of PN which would make it easier the modeling and design of distributed systems;
- a wide range of possible applications will be using the same representation which facilitate the re-use of modeling inside work domains;
- the expansion of reusability to cases among different work domains, reinforcing the use of PNs as a general schema;
- the extension of the use of Petri Nets beyond the modeling phase of design, including requirements analysis and validation.

Thus, it is very important to insert Timed Petri Nets in the proper context of the net standard, and in the context of PN extensions. During the last years a design environment has been built, in parallel with our study of Timed nets and its application to the design of automated and real time systems: the General Hierarchical Enhanced Net System (GHENeSys), where timed Petri Nets were included in a complementary way. That is, the time definition - which could be a proposal to part 3 of the standard - is made associating to each transition (place) a time interval, as proposed by Merlin [30] to model dense time. In the special case of deterministic transition (place) time it suffices to make the interval collapse by making the extremes equal to the same constant. For the case of a deterministic time PN, this imply in modifying Definition 6 to have the mapping $f : T \rightarrow \{\mathbb{R}^+ \times \mathbb{R}^+ \cup \{\infty\}\}$.

Besides the time extension, GHENeSys is also a hierarchical, object-oriented net which has also the following extended elements:

- **Gates:** which stands for elements propagating only information and preserving the marking in its original place. It could be an enabling gate, that is, one that send information if is marked or an inhibitor gate, if propagates information when is not marked. Of course GHENeSys does not allow internal gates. Thus gates should have always an original place, a special place called pseudo-box.
- **Pseudo-boxes:** denotes an observable condition that is not controlled by the modeled system. During the course of the modeling pseudo-boxes could also stand for control information external to the hierarchical components and could be collapsed when components are put together. Thus, pseudo-boxes must be considered in the structure of the net but should not affect its properties or the rank of the incidence matrix.

The graphic representation of the elements followed the schema shown in the Fig. 2 bellow,

Since our focus in this work is time extensions we illustrate hierarchy with a simple example net shown in the Fig. 3 Notice that hierarchical elements are such that the border is composed of only place or transition elements and has a unique entrance element and a unique output element. Besides, we require that each hierarchical element be simply live, that is, there is at least one live path from the entrance to the output. This is called a proper element in the theory of structured systems.

Definition 14.[GHENeSys] GHENeSys is tuple $G = (L, A, F, K, \Pi, C_0, \tau)$ where (L, A, F, K, Π) represents a net structure, C_0 is a set of multisets representing the initial marking, and τ is a function that maps time intervals to each element of the net.

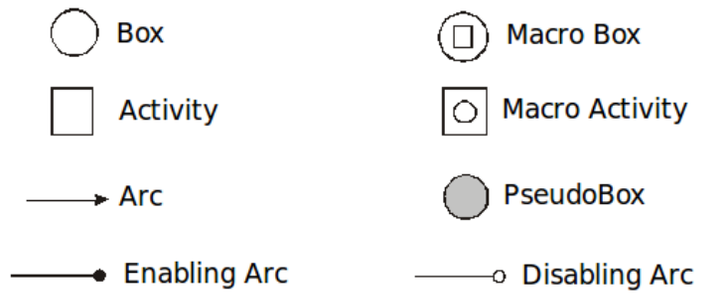

Figure 2. Graphic representation GHENeSys graphic elements.

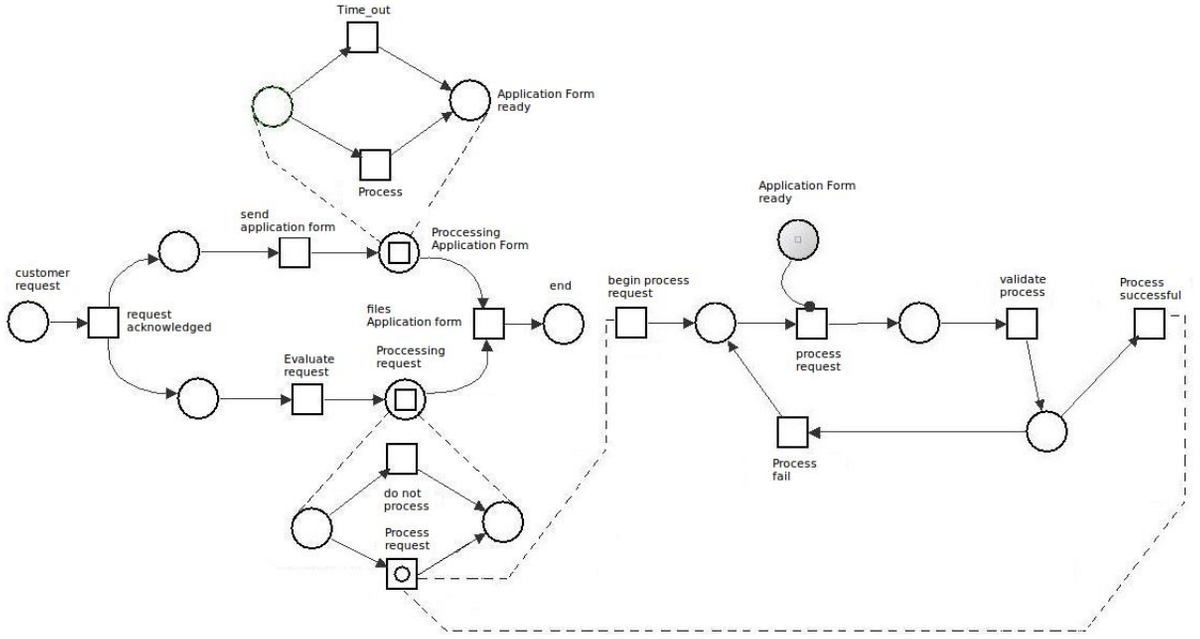

Figure 3. Example of hierarchical proper elements or macro-boxes and macro-transitions.

- $L = B \cup P$, are sets of places denoted by *Boxes* and *pseudo-boxes*;
- A is a set of activities;
- $F \subseteq (L \times A \to N) \cup (A \times L \to N)$ is the flux relation;
- $K : L \to N^+$ is a capacity function;
- $\Pi : (B \cup A) \to \{0, 1\}$ is a mapping that identifies the macro elements;
- $C_0 = \{(l, \sigma_j) | l \in L, \sigma_j \in R^+ \ |l| \leq K(l)\}$ is the marking of the initial state;
- $\tau : (B \cup A) \longrightarrow \{\mathbb{R}^+, \mathbb{R}^+ \cup \{\infty\}\}$ is a mapping that associates time intervals to each element of the net

3.1. A simple example of verification with GHENeSys

As mentioned before the main advantage of GHENeSys is to facilitate the verification of requirements and restrictions in the modeling and design of distributed systems. Therefore the environment should be able to related the elements and verify the interpretation of

formulas that could involve deterministic time (explicitly or not). In the simple example that follow we show how this verification is performed in the GHENeSys system.

Besides the illustration of the use of deterministic time and the GHENeSys net, the example also shows a method adopted to the modeling with Petri Nets, which is based on eliciting requirements in UML and then (if the target is a dynamic system) transforming the semantic diagrams of UML in classic Petri Nets[4]. A Petri Net with some extensions is created in the GHENeSys which also allows the insertion of formulas in CTL that can be verified. Figure 4 [5] shows the UML class diagram to this problem. In a cycle time three drives come to the station which has only two independent pumps.

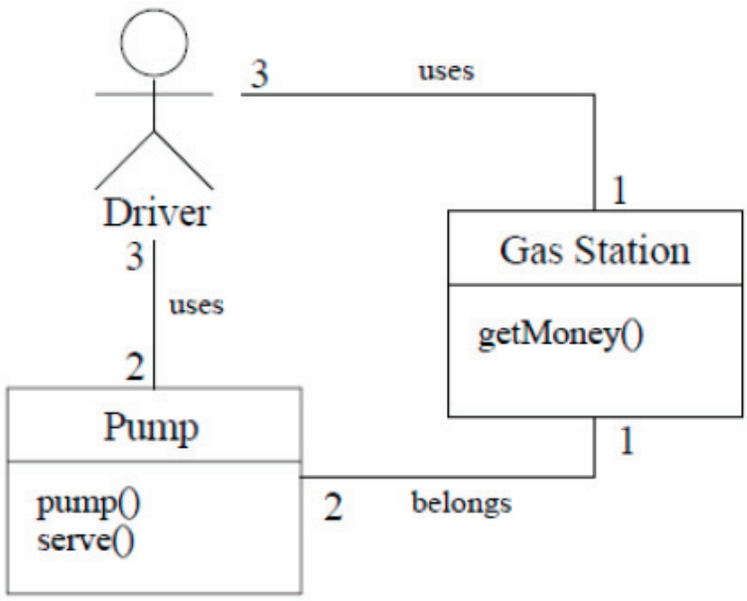

Figure 4. Class diagram to the problem of gas station.

In the gas station problem three different agents are identified: i) the gas station management who is responsible for charging the users, ii) the pumps that are supposed to serve gasoline to the costumers, and iii) the costumers, that is, drivers who are supposed to pay for a proper amount of gasoline and them help themselves. In this simple event we follow the model proposed in Baresi [5] where three drivers depends of only one cashier to pay for the gas and can use two different pumps to fill their cars. First of all we can guarantee that the proper process is followed and them we could insert a characteristic time in the basic operations. We used GHENeSys to provide the model using a classic P/T net. The resulting model is shown in the Fig. 5. This problem is to simple to use extensions but even in that case it would be possible to simply verify if the payment was done (using a gate) to enable the pump with the proper amount of gas instead or carrying the mark. For this problem it would be no significant difference in the size of the graph or in the resulting model.

The important feature here is to follow a modeling approach, which is implied in the steps described so far. Before modeling, requirements should be modeled in UML by semantic

[4] It would also be possible to synthesize a high level net, but this is not in the scope of the present work

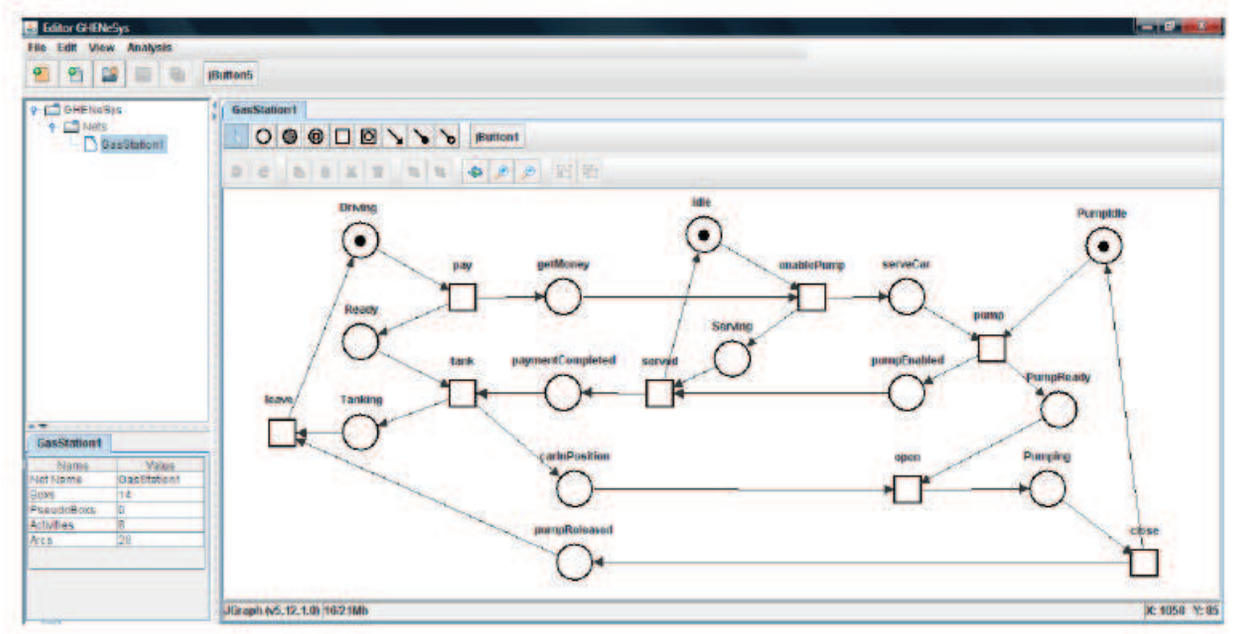

Figure 5. Classic model to the gas problem.

diagrams. There is a good discussion in the academy about the choice of the diagrams to each class of problem. Some authors prefer to go directly to SysML [4] while others just leave open the question about which diagrams should be used and invest in the analysis of this diagrams using Petri Nets [5, 10, 18, 39, 40].

Proceeding with our example let us suppose that we desire to verify some properties of the model such as

$$getMoney \longrightarrow \forall \diamond Pumping \tag{3}$$

$$\forall \Box (getMoney \longrightarrow \forall \diamond Pumping) \tag{4}$$

Using GHENeSys, formulas 3 and 4 can be evaluated by the Timed Petri Net modeling as we can see in the following.

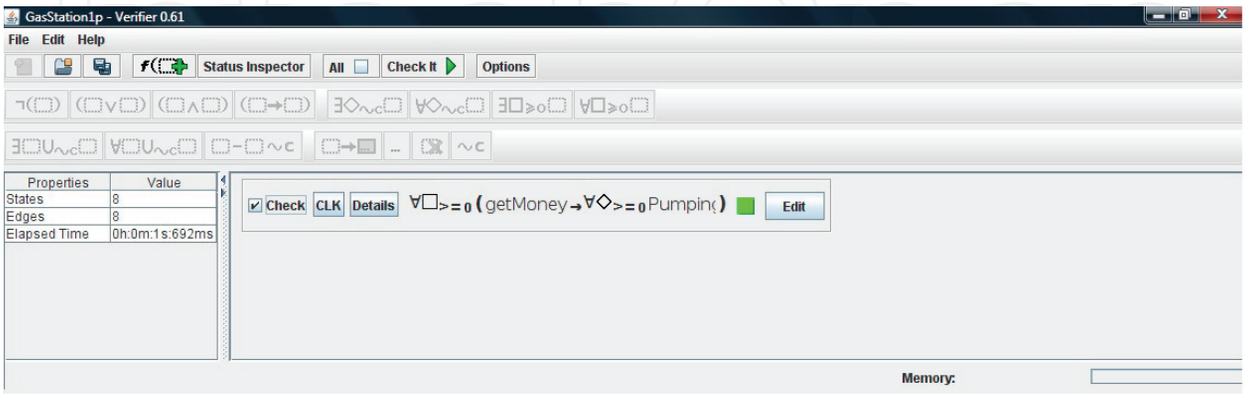

Figure 6. Sanpshot of the GHENeSys verifier for property 4.

The introduction of deterministic time (transition) would add more detail about the process, with the characteristic time for processing the payment or to fill a car. An organized queue would fail (even if works quite fine in the model) since this time can be modified depending of the user or to unpredictable events during the payment or during the supply process. However if specific (and deterministic)intervals such as 3 min for the payment and 5 min for the filling of gas are stablished, the system could handle 9 drivers in 25 min with a waiting time of at most 2 min for some drivers.

More convincing examples can be found in business, manufacturing or computer networks. More challenge problems emerged in the spatial applications or satellite control, but what is important is that even deterministic time approach can be used to solve a diversified set of problems. However, it could be stressed that the timed approach should be supported for tools and environments that rely in a sound and complementary approach to Timed Nets including Time Petri Nets. The approach shown here, inserted in the GHENeSys environment is exactly one of this cases. Besides, GHENeSys is an implementation of a unified net, that follows the specifications in ISO/IEC 15909 standard.

In the next section we go further in the discussion of using Petri Nets and specifically Timed Petri Nets to fit requirements that come in the new version of UML, which includes time diagrams and timelines.

4. PN as a general system representation framework

As pointed in the beginning of this work, Petri Nets has developed for the last fifty years to become a general schema for systems modeling.

In the previous section, we showed that a modeling discipline should be followed to achieve good results with Petri Nets formal representation, specially when time is an important variable to consider, either by deterministic time or using continuous dense time intervals. However, in the example above time does not appear explicitly at the beginning, since we started with the class diagram where there was no reference to duration time of the processes (supplying or payment). The problem them begins with a demand to a proper representation of time duration in UML that could later be transformed in a timed net.

To fit this demand UML 2.0 specification inserted an interaction diagram derived from the sequence diagram where time intervals or time duration are very important issues. Thus, once identified the actors and sub-systems in the model, their interaction could be viewed and modeled taking in account that it occurs during a running time where specific events can cause a change in the status of that interaction. Thus, the full relation can be described in what is called a state lifetime where several timelines show the evolution of the interacting components.

OMG (www.omg.org) shows a very appealing example of time diagram to model

In Figure 7 we can see a hierarchical superposition of levels and the action derived from the interaction between a user and a web system. Sub-systems invoked by this action and the time they spent to provide a proper action are explicitly depicted. As in the previous problem

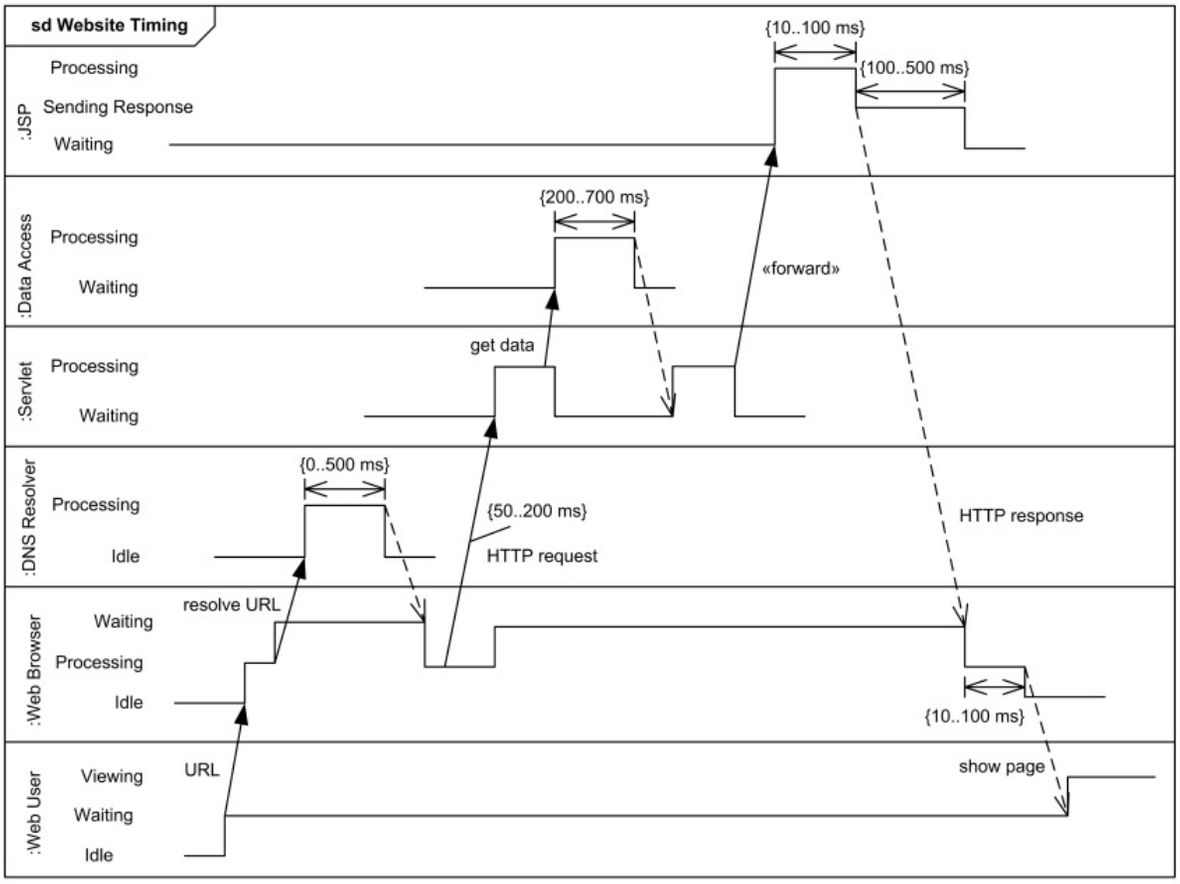

Figure 7. Time diagram for the web user latency.

of the gas station, the total time interval spent to serve one or nine users is the summation of the not superposed time intervals required for each dependent action.

Thus, the complete process would be to elicit the requirements using UML diagrams - including time diagrams - synthesize a Timed Petri Net from this model, and them perform the requirement analysis and final synthesis of a model for the problem. In fact, the final results for the example of the gas station were obtained following this approach.

Formally, the timelines are drawn according the behavior of *state variables*, defined in the following.

Definition 15.[State Variables] A state variable is a triple (V, T, D) where:

$V = v_i$ is a finite set of state values;

$T : V \to V$ specify each atomic transition or change in value.

$D : V \to \mathbb{N} \times \mathbb{N}$ is the time duration for each state value.

A timeline is the tracking of all changes in state value in a interval $[0, \tau)$ where τ is the observed time horizon. A timeline is said to be completely closed if the union of its not superposed values is exactly τ. In that case the transitions occur in deterministic time.

If the transitions occur in a time interval $[t_{min}, t_{max}]$ the timeline is said to be flexible. In this case we can represent the transition in a Timed Petri Net by an interval, as proposed in the first section. If we want to deal with deterministic time transition it is enough to make $t_{min} = t_{max}$ and the same net framework could be used.

Timeline models can be very useful in some critical problem applications such as intelligent planning and scheduling. Some of those applications could be used in spatial projects [14][5]. In other applications Petri Nets were used to perform requirements analysis including deterministic time, as in the one proposed by Vaquero et al.[40][41]. In that case the idea of solving real life planning problems starts with the elicitation and specification of requirements using UML, goes through the analysis of this requirements using Timed Petri Nets, synthesizes a model also in Petri Nets and finally uses a specific language, PDDL, to transfer the model to software planners which will provide the final result. Also, a modeling design environment were developed to perform this process[41][40].

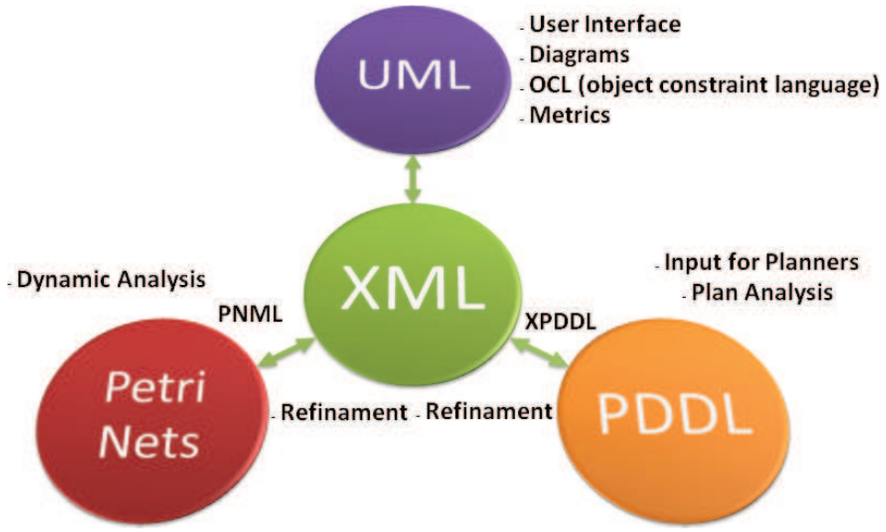

Figure 8. Language structure in itSIMPLE 3.1

A specific state lifetime were developed to model and analyze the timelines for the agents and objects that would compose the plan, as shown in Figure 8.

Based on this time diagram Petri Nets could be synthesized to make the proper validation of the model. It is important to notice that there are a large number of approaches and tools that claim to perform a good analysis of models directly associated to a planer software with good results. However, most of this systems address only model problems which are well behaved and/or have a limited size and complexity. When the challenge is to model a large system, such as the space project mentioned before or a port to get and deliver petroleum, the challenge could be too big to be faced by these proposals.

Therefore the combination UML/Timed Petri Nets could be successful in the modeling of large and complex problems also in the planning area, with the possibility to be applied in practice to real systems.

[5] See also the Mexar 2 Project and the use of intelligent software application in the link mexar.istc.cnr.it/mexar2.

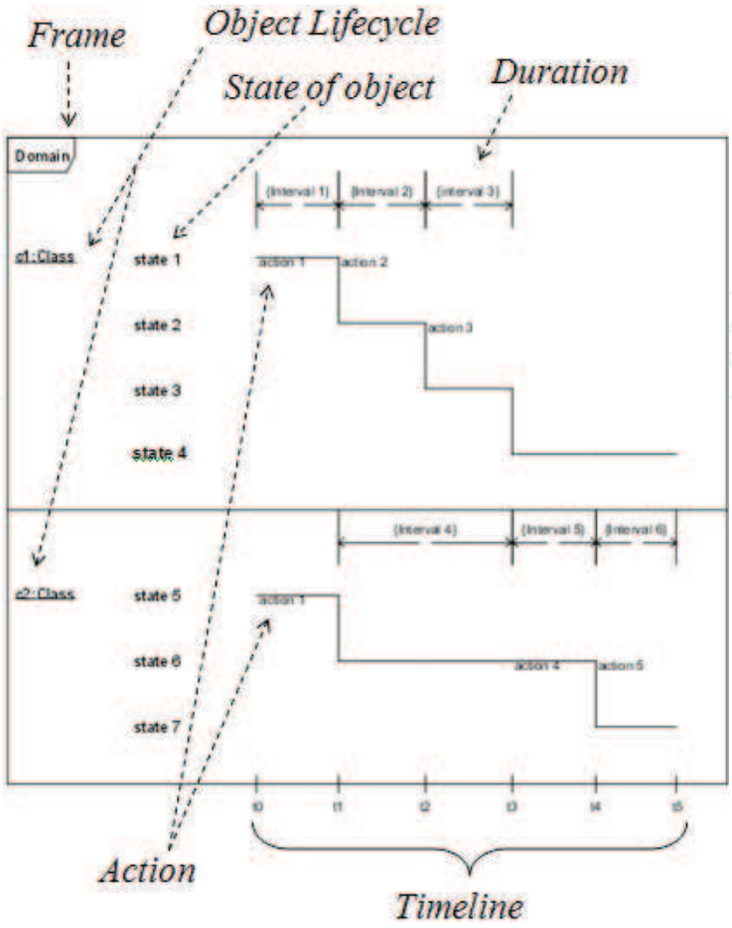

Figure 9. Time diagram in the itSIMPLE system.

5. Conclusions

In conclusion it is important to remark that the evolution of Petri Nets towards a formal representation, capable to treat complex systems should be based in two basis: the extension to model timed systems; and the development of a unified net that includes all extensions besides the timed approach - hierarchy, gates, not controled elements, always respecting the recent published ISO/IEC standard and its next release to appear in 2013. This is the fundamental concepts to have new environments that could support a complementary treatment of timed systems, that is, that could deal with deterministic timed net as well as with time PN in the same environment. That is the focus of the present work.

Besides, it would be advisable that the same environment could deal, also in a complementary way, with classic P/T nets as well as with high level (HLPN) nets or even with simetric nets, which is also part of the ISO/IEC standard. The novelty would be to use a unified net system as a platform to reach the further challenge which would be the introduction of abstract nets.

In what concerns the unified net to treat time intervals and dense time, we achieve a good point with the system GHENeSys where the present work focus most in the first part. However, in [17] a more detailed description of the state class algorithm is given and the basic concepts that lead to a modeling and simulation approach to dense time nets. Therefore the unification with timed PN is a promising result in the near future. Also the system GHENeSys

is being developed to implement a unified net as we described above, dealing with P/T and HLPN in the same environment. That is a good combination, capable to model and simulate timed and time nets (in that case using model checking) in the same environment, with the advantage to have a sound and formal representation supporting all process.

Thus, it would be possible to have performance analysis that really fits the complexity of the problem addressed, adapting very easily to discrete or dense time approach.

Acknowledgements

The authors acknowledge to all members of the Design Lab in Mechatronic Department for their work who are present in every part of this article be in the background, in the implementation of tools, in the search for new applications in the analysis of the problems already solved. Particularly we thank the work of Gustavo Costa, Arianna Salmon and Jose Armando S. P. Miralles.

Author details

Silva, José Reinaldo
University of São Paulo, Mechatronics Department, Escola Politécnica, São Paulo, Brazil

Del Foyo, Pedro M. G.
Federal University of Pernambuco, Department of Mechanical Engineering, Recife, Brazil

6. References

[1] Aalst, W van der (2002) Workflow Management: Models, Methods, and Systems, MIT Press, 365 p.
[2] Aalst, W van der (2004) Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Mangement , LNCS 3098 pp 1-65.
[3] Alur, R. and Dill, D. (1990) Automata for modeling real-time systems, *Lecture Notes in Computer Science* 443: 322–335.
[4] Andrade, E, Maciel, P, Callou, G, Nogueira, B, Araujo, C (2011) An Approach Based in Petri Nets for Requirements Analysis, in in Petri Nets Applications, Pauwel Pawlewski (ed.), InTech, 752 p.
[5] Baresi, L, Pezze, M (2001) Uniform Approaches to Graphical Process Specification Techniques, Electronic Notes in Theoretical Computer Science, vol. 44, pp. 107-119.
[6] Berthomieu, B. and Diaz, M. (1991). Modelling and verification of time dependent systems using time petri nets, *IEEE Trans. on Software Engineering* 17(3): 259–273.
[7] Berthomieu, B. and Menasche, M. (1983) . An enumerative approach for analyzing time Petri nets, *in* R. E. A. Mason (ed.), *Information Processing: proceedings of the IFIP congress 1983*, Vol. 9, Elsevier Science Publishers, Amsterdam, pp. 41–46.
[8] Berthomieu, B., Ribet, P. O. and Vernadat, F. (2004). The tool tina - construction of abstract state spaces for petri nets and time petri nets, *Int. J. Prod. res.* 42(14): 2741–2756.
[9] Berthomieu, B. and Vernadat, F. (2003). State class constructions for branching analysis of time petri nets, *Lecture Notes in Computer Science* 2619: 442–457.

[10] Bodbar, B, Giacomini, L, Holding, DJ (2000) UML and Petri Nets for the Design and Analysis of Distributed Systems, Proc. of the IEEE Int. Conf. on Control Applications, pp. 610-615.
[11] Boucheneb, H., Alger, U. and Berthelot, G. (1993) . Towards a simplified building of time petri nets reachability graph, *Proc. 5th International Workshop on Petri Nets and Performance Models*, Toulouse, France, pp. 46–55.
[12] Boucheneb, H. and Mullins, J. (2003). Analyse des réseaux temporels. calcul des classes en o(n2) et des temps de chemin en o(m x n), *Technique et Science Informatiques* 22(4): 435–459.
[13] Cerone, A. and Maggiolo-Schettini, A. (1999). Time-based expressivity of time petri nets for system specification, *Theoretical Computer Science* 216(1): 1–53.
[14] Cesta, A, Finzi, A, Fratini, S, Orlandini, A, Ronci, E (2010) Validation and verification issues in a temiline-bsed planning system, Knowledge Engineering Review, vol. 25, no. 3, pp. 299-318.
[15] Ciufudean C, Filote C (2010) Workflow Diagnosis Using Petri Net Charts, in Petri Nets Applications, Pauwel Pawlewski (ed.), InTech, 752 p.
[16] Daws, C., Olivero, A., Tripakis, S. and Yovine, S. (1995). The tool KRONOS, *Hybrid Systems III: Verification and Control*, Vol. 1066, Springer, Rutgers University, New Brunswick, NJ, USA, pp. 208–219.
[17] del Foyo, P. M. G. and Silva, J. R. (2011). Some issues in real-time systems verification using time petri nets, *Journal of the Braz. Soc. of Mech. Sci. & Eng.* XXXIII(4): 467–474.
[18] del Foyo, P. M. G., Salmon, A. Z. O., Silva, J. R. (2011) Requirements Analysis of Automated Projects Using UML/Petri Nets, Proc. of COBEM 2011.
[19] Girault, C, Valk, R (2003) Petri Nets for System Engineering, Springer, 607 p.
[20] Gardey, G., Lime, D., Magnin, M. and Roux, O. H. (2005). Romeo: A tool for analyzing time petri nets, *in* K. Etessami and S. Rajamani (eds), *CAV2005*, Vol. 3576, Springer-Verlag, pp. 418–423.
[21] Hadjidj, R. and Boucheneb, H. (2006). On-the-fly tctl model checking for time petri nets using state class graphs, *acsd* 0: 111–122.
[22] Hadjidj, R. and Boucheneb, H. (2008). Improving state class constructions for ctl* model checking of time petri nets, *STTT* 10(2): 167–184.
[23] ISO/IEC (2002) High-Level Petri Nets: Concepts, Definitions and Graphical Notation, International Standard Final Draft, ISO/IEC 15909.
[24] ISO/IEC (2005) Software and Systems Engineering - High-Level Petri Nets, Part 2: Transfer Format, International Standard WD ISO/IEC 15909.
[25] Kindler, E (2006) PNML: Concepts, Status and Future Directions, Invited paper, Proc. of EKA 2006, pp. 35-55.
[26] Kordic, V (ed.) (2008) Petri Nets: Theory and Applications, I-Tech Education and Pub, 208 p.
[27] Lafortune, S. and Cassandras, C. G. (2008). *Introduction to Discrete Event Systems*, second edn, Springer, 233 Spring Street, New York, NY 10013, USA.
[28] Larsen, K. G., Pettersson, P. and Yi, W. (2000). Uppaal - validation and verification of real time systems - status & developments.

[29] Marsan MA, Bobbio, A, Donatelli S (1998) Petri Nets in Performance Analysis: an Introduction, LNCS 1491, pp. 211-256.
[30] Merlin, P., Faber, D. (1976) Recoverability on communication protocols - implications of a theoretical study, IEEE Trans. on Communications, vol. 4, no. 9, pp. 1036-1043.
[31] Murata, T (1989) Petri Nets: Properties, Analysis and Applications, Proceedings of IEEE, vol. 77, pp 541-580.
[32] Patrice B (2010) A New Control Synthesis Approach of P-Time Petri Nets, in Petri Nets Applications, Pawel Pawlewski (ed.), InTech, 752 p.
[33] Pawlewski P (2010) Using Petri Nets to Model and Simulation Production Systems in Process Reengineering, in Petri Nets Applications, Pawel Pawlewski (ed.), InTech, 752 p.
[34] Ramchandani, C. (1973) Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, PHD thesis, MIT, 220 p.
[35] Riviere, N., Valette, R., Pradin-Chezalviel, B. and Ups, I. A. . (2001). Reachability and temporal conflicts in t-time petri nets, *PNPM '01: Proceedings of the 9th international Workshop on Petri Nets and Performance Models (PNPM'01)*, IEEE Computer Society, Washington, DC, USA, p. 229.
[36] Roux, O. H. and Déplanche, A. M. (2002) . A T-time petri net extension for real time-task scheduling modeling, *European Journal of Automation* 36: 973–987.
[37] Salmon, A. Z. O., Miralles, J. A. S. P., del Foyo, P. M. G., Silva, J. R. (2011) Towards a Unified View of Modeling and Design with GHENeSys, Proc. of COBEM 2011.
[38] Sifakis, J. (1980). Performance evaluation of systems using nets, *Proceedings of the Advanced Course on General Net Theory of Processes and Systems*, Springer-Verlag, London, UK, pp. 307–319.
[39] Tierry-Mieg, Y, Hilah, L-M (2008) UML Behavioral Consistency Checking Using Instantiable Petri Nets, Workshop UML in Formal Methods, 10th. Int. Conf. on Formal Engineering Methods.
[40] Vaquero, T.S., Silva, J.R., Ferreira, M., Tonidandel, F., Bech, J.C. (2009) From Requirements and Analysis to PDDL in itSIMPLE 3.0, Proc. of Int. Conf. in Artificial Planning and Scheduling, AAAI.
[41] Vaquero, T.S., Silva, J.R., Bech, J.C. (2011) A Brief Review of Tools and Methods for Knowledge Engineering for Planning and Scheduling, Proc. of Int. Conf. in Artificial Planning and Scheduling, AAAI.
[42] Wang, J (1998) Timed Petri Nets: Theory and Applications, Kluwer Academic Pub. 281 p.
[43] Wang, J, Deng, Y, Xu, G (2000) Reachability Analysis of Real-time Systems Using Timed Petri Nets, IEEE Trans. on Syst. Man and Cybernetics, vol 30 no. 5, pp. 725-736.
[44] Yoneda, T. and Ryuba, H. (1998). CTL model checking of time petri nets using geometric regions, *IEICE Trans. on Information and Systems* E81-D(3): 297–396.
[45] Zuberek, W. M. (1980). Timed petri nets and preliminary performance evaluation, *ISCA '80: Proceedings of the 7th annual symposium on Computer Architecture*, ACM Press, New York, NY, USA, pp. 88–96.