ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

INTRODUCING OBJECT-ORIENTATION IN UNIFIED PETRI NET APPROACH

Jose Reinaldo Silva, reinaldo@usp.br
José Armando San Pedro Miralles, jaspm2004@gmail.com
Dept. of Mechatronics and Mechanicals Systems, University of Sao Paulo, Brazil
Arianna Olivera Salmon, arianna@csd.uo.edu.cu
Computer Science Dept, Universidade de Oriente, Cuba
Pedro M. Gonzalez del Foyo, pedro.foyo@poli.usp.br
Dept. of Mechatronics and Mechanicals Systems, University of Sao Paulo, Brazil

Abstract. *Since the beginning of this century two major tendencies in the state-of-art of Petri Nets emerged: one is the confirmation of Petri Nets as a sound formalism to model several discrete systems in several areas of knowledge, since biological and chemical systems up to the control systems applied to manufacturing and automated process in general; another tendency is the introduction of a unified approach to Petri Nets, giving another perspective to the profusion of particular approaches and extensions. Even high-level nets are now envisaged as a general approach that subsumes elementary nets through the Basic High-Level Nets.*
However, besides the importance and adherence to the principles behind Petri Nets formalism, there is not a significant effort to provide practical tools that reflect that new view for Petri Nets, while still maintaining the same efficient to analyze, simulate and document preliminary design of distributed systems. Besides, even if several important papers on the subject also includes object-orientation as a feature of the unified approach, improving the use of Petri Nets in design, there is no attempt to go into the formal aspects of the relationship between object nets and high level nets as well as in the impact of this general net in practical software environments that support the modeling process.
In this work we present some partial results of a project called GHENeSys that proposes a unified object net that tries to encompass all features of a unified net that also includes object-orientation and provides a software environment for that (also called GHENeSys). This proposal is now evolving to include timed nets as described shortly in the final sections. A first version of the environment is used to simple workflow problems that requires a different set of properties of the modeling representation. That shows also the practical advantage of a unified system.

Keywords: *Petri Nets, unified approach, formal modeling, Discrete Systems*

1. INTRODUCTION

Petri Nets were originally created to represent communication processes in engineering and thus attached to a class of application. Further development make this initial approach to broad, while the formalism of nets were developed, based on graph theory and recently in type theory, sets, signatures, formal relations and restrictive filters. This broad sense of Petri Nets brought also a wide demand to apply these formal representation in several different areas, including some areas outside the scope of engineering. Complex discrete and distributed systems could them be developed since the very beginning using Petri Nets (PN), what raises the question of how to deploy the PN model to be analyzed and eventually simulated by a generic token player.

The question becomes a seek for a design approach based on PN instead of just proceedings and direct methods for a chosen application domain. Thus, it is mister to look for a general paradigm that associate the modeling initiative to a generic design process. Also, the broader use of PN also directs the use of these formalism in the specification and Requirement Engineering besides the conventional modeling and design phases. There are several works in the literature [Santos and Silva 2004] proposing the use of Petri Nets as a basic formalism to perform requirements analysis. Such approach had also the advantage of expressing requirements in the same formalism that would be used further in the design phase - at least for dynamic discrete systems.

Historically the first attempts to integrate PN and design approaches appear in the beginning of 80's where the basic idea was to Structured Analysis with PN [Miyagi 1988]. By this time the appealing discipline create in the project of numeric code, synthesized in SADT dominate the scene in Engineer and in computer programming. However, that proposed integration does not introduce any direct change in the formalism of Petri Nets. On the contrary, all the effort were directed to the construction of a process to synthesize a net, sometimes converted in a programming discipline [Miyagi et al 1988].

The structured approach succeed and was effectively used to solve several problems in discrete systems. Systems and programs continue to grow in size and complexity and other methods were researched such as the object oriented approach. Now the advantage came from a different kind of abstraction, called inheritance, and other features as polymorphism, encapsulation and a very convincing performance in reusability. All these properties are very useful to face nowadays discrete systems and to the applications of Petri Nets in business planning and manufacturing workflow, as well as in the

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

previous class of problems.

Object oriented Petri Nets appear just cover this gap, and a different discussion emerge with that: how to integrate Petri Nets with Object Oriented Design without loosing the formalism and good practices already consolidated in the discrete systems analysis. The other side of this question is how to make this same integration in a way to include the good features and features of the object oriented approach. Initially that discussion was translated in two different proposals: objects inside Petri Nets and Petri Nets inside objects. The first one tried to come with a net of objects, where the role of the net was to model the interaction among this objects. The second proposal defined the object net, that is, an object whose behavior was modeled by a PN. Other proposal appear in the literature even reinforcing one of the two directions - a net of objects or an object net - and other proposals faced the challenge of trying to find an unified approach. GHENeSys is one of them.

In the next sections we discuss the basis of this unified approach to integrate PN and object orientation, briefly comparing the options taken in the GHENeSys project with other work. The GHENeSys project is still in development but a public domain tool we developed and will be available in the site of Petri Nets World very soon.

2. OBJECT NETS

Since its origin in 1962 (in the doctoral thesis of Carl Adam Petri) Petri Nets have evolved fast in its formalization while expanding its area of application from the former communication systems to a very wide list of modeling domains. Therefore it was unavoidable the integration of PNs to the general methods of design of any discrete system or to any discrete approximation of a distributed system. Consequently, a good question is the integration of PNs in well established paradigms of design such as the object-oriented approach. In fact the question on “how to synthesize a Petri Net” is dated from the beginning of the 80’s and early approaches proposed an integration with structured methods such as the SADT [Miyagi et al 1988], the main approach in that period. Following, all attention was turned to the integration between Petri Net approach and object-orientation.

During the 90’s several object-oriented Petri nets(OO-nets) were proposed [Buch and Guelfi 2000], [Lomazova and Schnoebelen 2000] [Rumbaugh et al 1991], [Lakos 1995]. Basic conceptual questions were the introduction of polymorphism and the recursive nature of classes, since an important characteristic of OO-nets is that a net can be in another net as a token (nets-within-nets paradigm) [Valk 2004]. Such extensions are helpful for modeling hierarchical multi-agent distributed systems. Lakos et al. modeled network protocols with OO-nets [Lakos et al 1995]. Moldt and Valk (2000) proposed Bussines Process Petri nets to model workflows by OO-nets opening a possibility to put together application design and business process.

However, he real challenge was to discover a proper way to integrate a generative synthesis paradigm for design, such as object-orientation, with relational formal presentation originally designed for communication processes. Even if we consider the late development of Petri Net formalism as a general modeling approach, a question remains on “how a net model would be synthesized, particularly to complex systems”. The proposed fusion with object-orientation could answer that question. On the other hand, this fusion is also a challenge. Thus, unified nets and problem of synthesis and documentation of systems were not seen as different aspects of the same problem.

The Introduction of restrictions in the object properties, such as the restriction in the object structure to single inheritance was the first step towards a general approach to net of object nets. Another interesting topic is to preserve hierarchy which was already present even in the colored nets as proposed in Design CPN (reference).

The first version of the GHENeSys (General Hierarchical Enhanced Net System) system faced the challenge of introducing a simple implementation for the net of object nets, which could instantiate most of the extend nets, most of which were associated to specific application environments. A good result for that was the definition of a state equation and the possibility to analyze behavioral and structural properties [del Foyo and Silva 2003] where some promising results that allow pursue some advances, introducing a higher level approach by adding a new class in the system: the token.

Making the token an object can solve two problems at the same time: providing distinguishable marks for the net by introducing attributes to the tokens, what we will analyze in the next section.

3. A PROSPECTIVE RELATION BETWEEN OO-PETRI NETS AND HIGH LEVEL NETS

The original concern in the synthesis of nets evolve from the structured approach proposed in the 80’s by Miyagi et al (1988) was revisited to provide the object-oriented synthesis as object nets or, in other words, a object description where the objects have its behavior represented by processes described in Petri Nets. Naturally the challenge evolve to compose a network of object nets and obtain a recursive approach. Several articles appear in the literature to cover one or other approach and very few faced the problem of covering both. As in the development of classic Petri Nets, most of these proposals were mixed with the commitment of attending a specific class of applications, and the need to develop a new approach (or a hybrid one) were not so clear.

A representative work that integrates higher level nets - colored Petri Nets, in fact - with object orientation and with the concern of representing the knowledge associated with the synthesis of the net was proposed by Bañares et al (2001).

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

Maier and Moldt (2001) also proposed the OCP-Net (Object Colored Petri) where a similar merging were envisaged, even without the charge of representing direct knowledge. The hole of structure and hierarchy was clarified by Guerreiro et al (2001) that proposed a modular approach over an object-based CPN (Colored Petri Net). A more sound approach to object nets with a behavior described by Petri Nets was proposed in [Bae and Hong 2001] but that still miss a system approach to the object community, even if it is suited to represent multiagent systems. Finally Miyamoto (2005) present a survey based on methods that try to superpose two different levels using object nets as tokens. Unfortunately that also make the formalism necessary to describe the meta-model complex even if the not trivial link with the object nets behavior.

We claim that a recursive approach is easier and clear to those interested in the design of distributed system, independently of the nature of the application, if requirements analysis, design rationales, workflow management in manufacturing environments, business processes or communication between processes. That is what we call GHENeSys.

The proposal of GHENeSys is based on a unique recursive and integrated definition of a object oriented Petri Net, were each component is an object net as well as the whole net. Composing the basic class definition we have not only the box class (passive elements) and the activity class (dynamic elements) there is also a class token, to which we associate only attributes. Therefore, there is no second level formalism and the overall schema is pretty close to a colored net. The advantage is to have a complementary proposal that relies on the previous established formalism of colored nets, conventional place transition nets and basic high level nets. Besides the object approach to the net is also present.

As a result, the design discipline reinforced by this approach is a systemic one, based on the dynamic relationship among the composing elements instead of a static one. At beginning of the design process it is necessary to have what we call a plan, that is, a general model of the the basic dynamic relations among the basic object classes. The remaining of the process should come as a combination of refinement (using hierarchy and direct abstraction) and (single) inheritance, what is a simple way to preserve the scope of the process.

There is also a need to integrate token flow with message passing, what is introduced by the introduction of special passive elements belonging to the box class, called pseudo-box, combined with the use of gates. Thus, the persistent marking of the pseudo-boxes is guaranteed by the gates, through which only information will flow.

In the next section we show GHENeSys definitions and derive the basic state equation. The class diagram is also showed and all the implications of the integrated approach discussed in more detail.

4. THE GHENESYS PROPOSAL

In recent years has arisen an idea of an unified approach that could encompass all the available resources found within the different Petri nets extensions which are necessary to deal with most, if not all, of discrete event systems(DES). Despite the increasingly growing number of researchers in the field that considers that this kind of formalism is possible and desirable, yet no practical tool has been developed in this direction.

The net called *GHENeSys* was designed and developed in DLab from ideas presented in [Miyagi 1988], [Silva and Miyagi 1995], [Silva and Miyagi 1996] [Silva 1998], and finally formalized in [del Foyo 2001]. Initially, it was conceived as an extended net with concepts of object-orientation, a mechanism of abstraction -through the definition of hierarchy-, and synthesis -using a structured approach. *GHENeSys* have a great potential to become in that tool capable of represent, in a unified way, Petri nets and their extensions, as well as the high level Petri nets.

4.1 GHENeSys Definition

At the current evolution stage, *GHENeSys* (General Enhanced Hierarchical Net System) can represent all the enhanced nets cited in the literature. It was also modified to allow the representation of timed systems.

Definition 1 (GHENeSys). *GHENeSys* is a tuple $G = (L, A, F, K, \Pi, C_0, \tau)$ where (L, A, F, K, Π) represents the net structure, C_0 is the set of tokens on the initial marking, and τ is a mapping function, that maps the time intervals of each element of the net.

- $L = B \cup P$, are the places, them can be named *Boxes* and *PseudoBoxes* (this can be enabling or inhibitors);
- A are the activities;
- $F \subseteq (L \times A \rightarrow N^+) \cup (A \times L \rightarrow N^+)$ is the flow relation;
- $K : L \rightarrow N^+$ is the capacity function;
- $\Pi : (B \cup A) \rightarrow 0, 1$ is the function wich identifies the macro elements;
- $C_0 = \{(l, \sigma_j) | l \in L, \sigma_j \in R^+ \ |l| \leq K(l)\}$ is the set of tokens on the initial marking;
- $\tau : (B \cup A) \longrightarrow \{Q^+, Q^+ \cup \{\infty\}\}$ is a function that maps the time intervals of each element of the net.

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

The set of tokens is formed by the pairs (l, σ_j) where $l \in L$ determines the place where the token is located and σ_j is the clock wich measures the time that tokens remains on that place. This clock is updated everytime that a transition happens and the clocks of all tokens are sincronized with a global clock. As there may be several tokens in one place, the set of tokens is in fact a *multi-set*.

The function τ maps one time interval with each element of type *Box* or *Activity*. Don't have any sense to put time in *pseudoboxes* and tokens related to these, because these are elements with constant marking, i.e. the marking of *pseudoboxes* is not changed by the dynamics of the net[1]. The time intervals related to the net elements have different semantics depending on the type of element to which they belong. For example, for *boxes* the time interval represents the minimun or maximun amount of time that a token can remains in that element. For the *activities*, the time interval have the same semantic that have in Merlin's TPN transitions.

Time intervals are formed by their lower and upper limits. For each net element $e \in B \cup A$, the time intervals will be set as $[\downarrow e, \uparrow e]$ with $\downarrow e \in Q^+$ and $\uparrow e \in Q^+ \cup \{\infty\}$.

The flow relationship F was also modified to allow weight in the arcs, like in place/transition nets and TPN. The rest of definition items keep the same meaning they have in its original definition in [del Foyo 2001].

4.2 GHENeSys as an OO-Petri Net

Besides the introduction of structure in the modeling process, the *GHENeSys* proposal also aims to summarize the various possibilities of extend its formalism, ussing a object-oriented approach starting with the generic classes *Place* and *Activity*.

As in other proposed works, the *GHENeSys* elements are objects, and tokens are passive objects(contains no methods). The net represents the structure of the control system, while the tokens models the data structure of the system, but this data do not appear in the transition firing rules and therefore do not change radically the formalism of the net, such as Pr/T nets and coloured nets Petri nets [del Foyo 2001].

The net elements that belong to a particular class, can have subnets connected to them. In case of these elements represents structures that are widely known and frequently used in models, the subnets that they represents can be replaced by class methods. An example of this case is the application of a discipline to manage the arrival of elements in a box, introducing a FIFO approach, used in various extensions that distinguish tokens by tags, to model for example, flow of messages in networks and the Internet. These methods represents some very well known behaviors and does not alter the properties of the net, such as FIFO and FILO buffers. In this way is possible to reuse these definitions in various models, as well as to easily expand these concepts through the mechanism of inheritance.

The objects attributes of *GHENeSys* can also be used to introduce the concept of time(discrete) in the net formalism as we will show later.

GHENeSys was defined in the previous subsection as a tuple $G = (L, A, F, K, \Pi, C_0, \tau)$, according to definition 1. The sets *L* and *A* are represented by the classes *Place* and *Activity* respectively. Therefore, the elements of set *L* are all objects of *Place* class and all the elements of set *A* are objects of *Activity* class. As in a system of classes, each one of these objects inherits the properties and methods of the class to which they belong. In *GHENeSys*, both *Place* and *Activity* classes have subclasses which allows to represent different elements, including compounds elements.

Therefore, *GHENeSys* is a cluster (a system composed by objects of different classes) and not a superclass as is stated in other proposals. In this approach, there is also a unification of "Objects within a Petri net" and "Petri nets within objects" trends, but without affecting the nets formalims. In this proposal the net is composed by objects that belongs to two classes and each is related with objects of the opposite class, maintaining the bipartite characteristic of Petri nets. These objects may contain subnets, recalling that those subnets must respects the definition of macro element.

Another advantage of the object orientation that is used in this proposal is the possibility of put in the methods, the behavior of elements already known or treated in other models which characterizes the re-use. This can be done when the structure and the net firing rule are respected. In principle, this can be seen as a limitation, and is indeed, but that ensure the consistency of the formalism. Despite this limitation, which is considered essential, the introduction of these elements constitutes a step forward because it allows to encapsulate and reuse elements and situations that usually appear in models, and that helps to reduce the problem of state explosion, a common problem founded when medium and large scale real-life system are modeled.

Another fundamental difference, in terms of systems analysis, of this proposal in relation to others, is the fact that for any level of abstraction, we have a net structure to which it can be applied all the known elementary analysis techniques of Petri nets.

The class diagram used in *GHENeSys* implementation is shown in figure **??**.

[1] there are special cases where the marking of *pseudoboxes* is modified but this will be discused later.

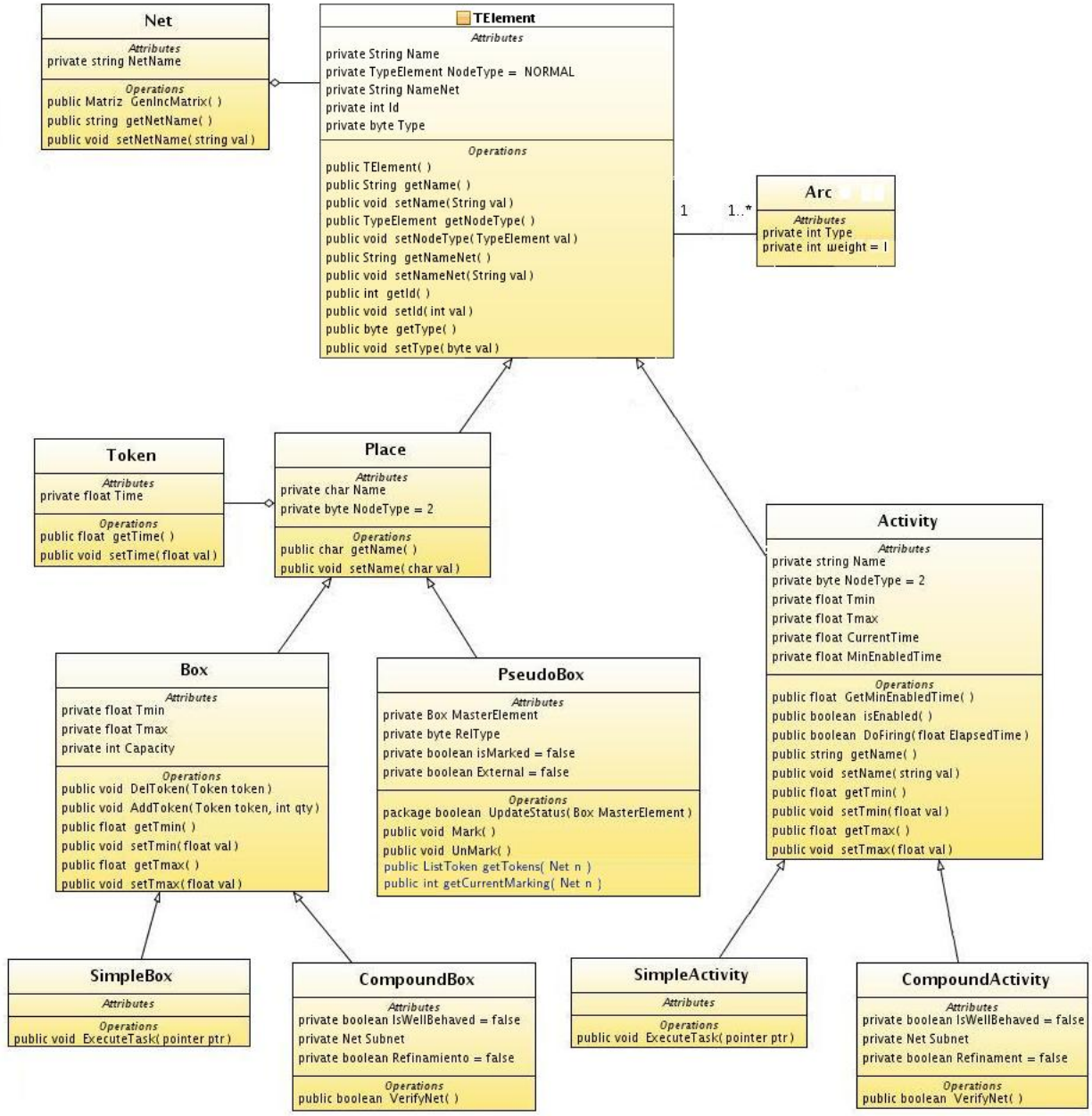

Figure 1. The GHENeSys Class Diagram

5. THE NEW FEATURES

Previus stages in *GHENeSys* evolution, treated time issues in Ramchandani's way [Ramchandani 1974]. Taking into consideration that most processes are not deterministic, like in real-time systems, such approach was modified. The current temporal approach in *GHENeSys* let us represent non deterministic durations like in Merlin temporal approach [Merlin and Faber 1976]. The approach used in *GHENeSys* allow the hierarchy levels representations even for passive elements wich is a very useful issue to deal with complex systems.

5.1 States and State Transitions in *GHENeSys*

Before introducing the definitions of state and state transitions we will present the definition of *multi-set* that will be used to represent the tokens in *GHENeSys*.

Definition 2 (Multi-set). A *multi-set* b, over a set A, its a function of A in N, $b : A \rightarrow N$. If $a \in A$ then $b(a)$ is the number of ocurrences of the element a on the *multi-set* b. A_{MS} is the set of all *multi-sets* under A. The empty *multi-set*

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

is denoted by $\varnothing$.

Definition 3 (State). A state in net G is a pair $s = (M, \sigma)$ where M is areachable marking in G and $\sigma : A \rightarrow Q^+$ is a vector which contain a timer for each enabled activity by marking M related to the time in which activity became last enabled. We denote as $\sigma(a)$ the timer for activity $a \in A$.

Now that the definition of state was established we can define the enabling condition for an activity in *GHENeSys*.

Definition 4 (Enabling Condition). An activity $a \in A$ is enabled at state $s = (M, \sigma)$ with $s \in R(s_0)$, $M = \eta(C_t)$ and $x \in R^+$ if the next conditions are satisfied:

$$\forall l_i \in {}^\bullet a \quad , \quad (l_i, x)(C_t) \geq n[l_i, a]; \tag{1}$$

$$\forall l_i \in a^\bullet \quad , \quad m_i \leq K(l_i) - n[a, l_i]; \tag{2}$$

where $n[l_i, a]$ is the weight of the arc which conect place l_i to activity a and $n[a, l_i]$ is the weight of the arc which conect activity a to place l_i. The set of all enabled activities in state s is denoted as $enb(s)$.

Condition **??** checks the availability of tokens in pre-conditions of activity a and condition **??** checks the capacity availability in post-conditions of a.

For an activity to fire, all tokens which enabled the activity must have its timers in zero. Then in order to determine if such enabled activity can be fired or not the minimum enabled time must be determined.

Definition 5 (Minimum Enabled Time). The Minimum Enabled Time for an activity a is the minimum time elapsed to reset the timers which enabled such activity. We denoted as ν_s the vector which contain the minimum enabled time for each activity enabled in s.

$\nu_s : a \rightarrow R^+ \quad \forall a \in enb(s)$.

The firing condition can be obtained from the enabling condition, the static time interval and the value of the activity timer for each enabled activity in s.

Definition 6 (Firing Condition). An activity $a \in A$ can be fired in state $s = (M, \sigma)$ iff:

$$a \in enb(s) \quad ;$$

$$\forall a_i \in enb(s),\ a_i \neq a \quad , \quad \downarrow a + \nu_s(a) - \sigma(a) \leq \uparrow a_i + \nu_s(a_i) - \sigma(a_i)$$

The set of all activities firable in s is denoted by Υ_s, $\Upsilon_s \subseteq enb(s)$.

Definition 7 (State Transition) Firing activity $a \in \Upsilon_s$ lead to an state transition $(M_1, \sigma_1) \xrightarrow{a,\delta} (M_2, \sigma_2)$ in time $\delta \in R^+$, such as $\downarrow a + \nu_s(a) - \sigma_1(a) \leq \delta \leq \uparrow a_i + \nu_s(a_i) - \sigma_1(a_i), \forall a_i \in enb(s_1)$ trought the next modifications:

$$\begin{aligned}
\forall (l_i, x_j) \in C_{t1} \quad &, \quad C_{t1} = C_{t1} \setminus \{(l_i, x_j)\} \cup \{(l_i, max(0, x_j - \delta))\};\\
\forall l_i \in {}^\bullet a \cap B \quad &, \quad C_{t2} = C_{t1} \setminus n[l_i, a] \times \{(l_i, 0)\};\\
\forall l_i \in a^\bullet \cap B \quad &, \quad C_{t2} = C_{t1} \cup n[a, l_i] \times \{(l_i, \downarrow l_i)\};\\
&\quad M_2 = \eta(C_{t2});\\
\forall a_k \in enb(s_2) \quad &, \quad \sigma_2(a_k) = 0;\\
\forall a_k \in enb(s_1) \cap enb(s_2), a_k \neq a \quad &, \quad \sigma_2(a_k) = \sigma_1 + \delta - \nu_s(a_k);\\
&\quad \sigma = \sigma + \delta;
\end{aligned}$$

$s \xrightarrow{a,\delta} s'$ define the set of states s' reachables from state s firing activity a in time δ, i.e., $|s'| = |\Upsilon_s|$.

Notice that for passive elements only one value of the time interval is used in Definition 7. That is because the created tokens inhered the maximum or minimum time interval depending on the semantic previusly defined.

$$\forall l_i \in L \quad \sigma_{li} = \begin{cases} \downarrow b & \forall l_i \in B \text{ when the minimal semantic is used} \\ \uparrow b & \forall l_i \in B \text{ when the maximal semantic is used} \\ 0 & \forall l_i \in P \end{cases}$$

Because the time interval attached to passive elements in *GHENeSys* the hieralchical concept is preserved even for the most abstracts levels. Of course the limitation in using only one of the time interval limits allow either the best or worst temporal scenarios determination at the time.

According to the identified semantics for firing activities in [Riviere et al. 2001], *GHENeSys* also led to choose between the strong and weak semantics. The default semantics applyed in *GHENeSys* are the minimal and strong semantics.

Definition 7 estipules the actions to be execute to compute states transitions. Using such definitions the reachability tree can be computed from some initial state.

5.2 A small example

Figure **??** show an example of GHENeSys net.

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

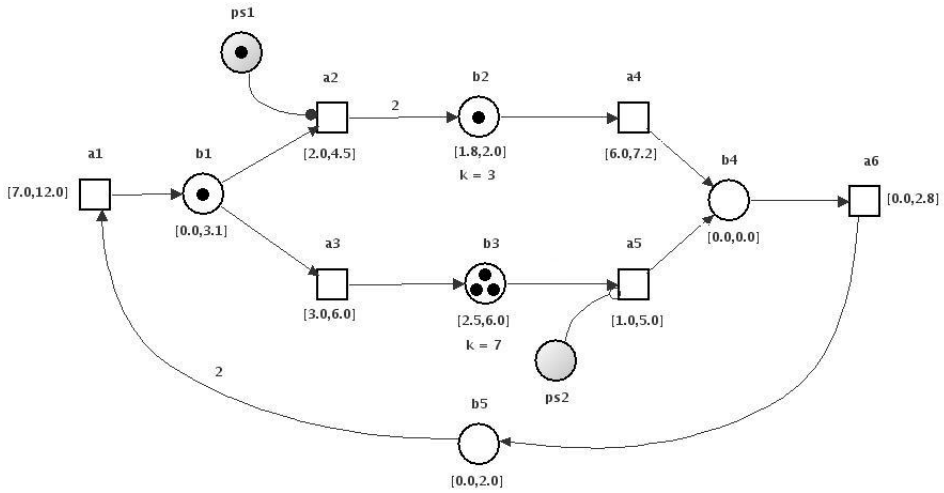

Figure 2. A small example of a GHENeSys net

The initial marking in such net is represented by the following multi-set,

$$C_0 = \{(1, 0.3), (2, 1.8), (3, 2.5), (3, 2.5), (3, 2.5), (6, 0)\}$$

From that initial marking the enabled activities, the minimum enabled time and the firing interval for each firable activity is computed. The results obtained are showed in the next table

$enb(s_0) =$	a_2	a_3	a_4	a_5
$\nu_{s_0} =$	0.3	0.3	1.8	2.5
δ	[2.3 4.8]	[3.3 4.8]	[7.8 9]	[3.5 4.8]

Recalling that the initial marking vector is $M_0 = [1\ 1\ 3\ 0\ 0\ 1\ 0]^T$, and all activities timers are reseted $\sigma_0 = [0\ 0\ 0\ 0]^T$, firing the activity a_2 with $\delta = 2.4$ at $s_0 = (M_0, \sigma_0)$ will lead to the marking $s_1 = (M_1, \sigma_1)$. The state transition is represented by:

$$(M_0, \sigma_0) \xrightarrow{a_2, 2.4} (M_1, \sigma_1)$$

Following Definition 7, the reached state s_1 will be:

$$\begin{aligned}
M_1 &= [0\ 3\ 3\ 0\ 0\ 1\ 0]^T \\
\sigma_1 &= [0.6\ 0]^T, enb(s_1) = \{a_4, a_5\} \\
C_1 &= \{(2, 0), (2, 1.8), (2, 1.8), (3, 0.1), (3, 0.1), (3, 0.1), (6, 0)\} \\
\sigma_1 &= [0.6\ 0]^T \\
enb(s_1) &= \{a_4, a_5\} \\
\Upsilon_{s_1} &= \{a_5\}
\end{aligned}$$

Note that the timer of activity a_4, $\sigma_1(a_4)$ is in 0.6 time units but even in such case, only activity a_5 will be able to fire because its latest firing time is less than the early firing time of a_4 even considering $\sigma_1(a_4)$.

6. CONCLUSIONS AND FURTHER WORK

In conclusion, we should remark that a real integration between the formalism of Petri Nets and object orientation turns out in an expressive formal representation that encompass all the application domains modeled by Petri Nets plus a set of new domains that demands a complete design process, including the requirements analysis. The possibility of representing rationales would improve a lot and the possibility of including time would also amplify the range to include real time applications.

The possibility of unifying the net in a general environment like GHENeSys would also open the possibilities to represent in the same environment single components, complex tools and integrate all of them in a suitable architecture to compose the overall system. The composition between hierarchy with inheritance will allow the combination of structuring and classification taking the advantages of both: an organized refinement procedure during design, and the reusability and conservative evolution.

However, the lack of a consensus formalism for objects has a great impact in this association since it also multiplies the possibilities for the association objects and Petri Nets. Object nets appears to be more attractive for some authors while

others, like the authors of this work, prefer a more direct recursive approach. The disadvantage of this last approach is that the composed formalism must be rebuilt from the very beginning. On the other hand, the interpretation of components and its relation would be facilitated. Also, a considerable part of the net formalism is preserved, even if some modifications would be necessary. The test of the general GHENeSys tools has not been thoroughly done, specially in what concerns the complexity of real applications. Therefore, partial results, including those with timed systems are very promising and we believe that it will proved to be very useful in the area of requirements analysis, information systems, workflow, home and building automation, scheduling, planning of huge systems and other similar systems.

For further work, besides the publishing of the of a tool with the development just showed in this work, we plan introduce some algorithms for property analysis, specially invariant synchronic distance and formal verification procedures. The treatment of rationales would also deserve attention since the modern design would require a good and sound documentation.

Another direction for research is to use a tool like GHENeSys as an early process to synthesize the implementation of PLC programs taking a graficet approach as an intermediary step. That could be done just with the classical part of the unified environment but could open a new great domain of applications.

7. REFERENCES

D. H. Bae and J. E. Hong. High-level petri net for incremental analysis of object-oriented system requirements. *IEE Proceedings of Software*, 148:11–18, 2001.

Didier Buchs and Nicolas Guelfi. A formal specification framework for object-oriented distributed systems. *IEEE Trans. Softw. Eng.*, 26(7):635–652, 2000.

P. M. G. del Foyo. Ghenesys: Uma rede estendida orientada a objetos para projeto de sistemas discretos. Master's thesis, Escola Politecnica da Universidade de São Paulo, 2001.

P. M. G. del Foyo and J.R. Silva. Towards a unified view of petri nets and object oriented modeling. *ABCM Symposium Series in Mechatronics*, 1:518–524, 2003.

Dalton Serey Guerrero, Jorge C. A. de Figueiredo, and Angelo Perkusich. An object-based modular cpn approach: Its application to the specification of a cooperative editing environment. *Lecture Notes in Computer Science*, pages 338–354, 2001.

C. A. Lakos. From coloured petri nets to object petri nets. In *Proceedings of the Application and Theory of Petri Nets 1995*, volume 935, pages 278–297. Springer-Verlag, Berlin, Germany, 1995.

Charles A. Lakos, John Lamp, Chris Keen, and Brian Marriott. Modelling network protocols with object petri nets. In *Workshop on Petri Nets Applied to Protocols*, pages 31–42, 1995.

Irina A. Lomazova and Ph. Schnoebelen. Some decidability results for nested petri nets. In *PSI '99: Proceedings of the Third International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 208–220, London, UK, 2000. Springer-Verlag.

Cristoph Maier and Daniel Moldt. Object coloured petri nets - a formal technique for object oriented modelling. *Lecture Notes in Computer Science*, pages 406–427, 2001.

P. Merlin and D. Faber. Recoverability of communication protocols–implications of a theoretical study. *IEEE Transactions on Communications [legacy, pre-1988]*, 24(9):1036–1043, Sep 1976.

Paulo Eigi Miyagi. *Control System Design, Analysis and Implementation of Discrete Event Production Systems by using Mark Flow Graph*. PhD thesis, Tokyo Institute of Technology, Tokyo, 1988.

Toshiyuki Miyamoto and Sadatoshi Kumagai. A survey of object-oriented petri nets and analysis methods. In *IEICE TRANS. FUNDAMENTALS, VOL.E88-A, NO.11*, 2005.

Daniel Moldt and Rudiger Valk. Object oriented petri nets in business process modeling. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 254–273, London, UK, 2000. Springer-Verlag.

C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

N Riviere, R. Valette, B. Pradin-Chezalviel, and I. A. Ups. Reachability and temporal conflicts in t-time petri nets. In: *Proceedings of the 9th international Workshop on Petri Nets and Performance Models*. Washington, DC, USA: IEEE Computer Society, 2001. p. 229.

James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-oriented modeling and design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

Eston A. dos Santos and J. R. Silva. Appying petri nets to requirements validation. *ABCM Symposium Series*, 1:508–517, 2004.

Jose Reinaldo Silva. Interactive design of integrated systems. In Luis Camarinha Matos and Hamideh Afsarmanesh, editors, *Intelligent Systems for Manufacturing*. Kluwer Academic Pub., 1998.

Jose Reinaldo Silva and Paulo Eigi Miyagi. PFS/MFG: A high level net for the modeling of discrete manufacturing systems. In Luis Camarinha Matos and Hamideh Afsarmanesh, editors, *Balanced Automation Systems, Arquitectures*

ABCM Symposium Series in Mechatronics - Vol. 4 - pp.451-459

and Design Methods, pages 349–362. Chapman & Hall, 1995.

Jose Reinaldo Silva and Paulo Eigi Miyagi. A formal approach to pfs/mfg: a petri net representation of discrete manufacturing systems. In *Studies in Informatics and Control*, Romenia, 1996. IC Publications.

P. E. Miyagi; K. Hasegawa; K. Takahashi. A programming language for discrete production systems based on production flow schema and mark flow graph. *Transactions of SICE*, (24):183–190, 1988.

Rudiger Valk. Object petri nets: Using the nets-within-nets paradigm. In Grzegorz Rozenberg (Eds.) Jorg Desel, Wolfgang Reisig, editor, *Lecture Notes in Computer Science*, volume 3098, pages 819–848. Springer-Verlag, June 2004. InternalNote: Submitted by: hr.

J. A. Bañares; P. R. Muro-Medrano; J. L. Villarroel; F. J. Zarazaga. Kron: Knowledge engineering approach based on the integration of cpns with objects. *Lecture Notes in Computer Science*, pages 355–374, 2001.

8. Responsibility notice

The author(s) is (are) the only responsible for the printed material included in this paper