A MONITORING APPROACH FOR DISCRETE EVENT SYSTEMS BASED ON A TIME PETRI NET MODEL

Mohamed Ghazel * **Armand Toguyéni** *
Michel Bigand **

* *Laboratoire d'Automatique, Génie Informatique et Signal*
** *Équipe de Recherche en Génie Industriel*
École Centrale de Lille, BP 48, 59651 Villeneuve d'Ascq, France

Abstract: We develop in this paper a monitoring approach for Discrete Event Systems (DES) starting from a time Petri net model representing the a priori known behavior of such a system. The originality of our approach lies in the combination made of the concept of event observability with the exploitation of the temporal constraints on these events in order to refine the result of the monitoring process.

Keywords: Monitoring, diagnosis, time Petri nets, discret event system, time.

1. INTRODUCTION

In this paper, we propose a monitoring approach to be applied to Discrete Event Systems (DES) for which one knows the behavior a priori. This behavior is represented by a Time Petri Net model (TPN hereafter). In addition, the events which can occur are of two types: observable and unobservable. Our objective here is to develop a method which allows filling up this partial observability on the system in order to track online its state and to identify the events which occur. That will mainly enable to monitor the system by discerning online possible failures. The monitoring approach that we propose uses first the observable events to *estimate* the states the system can assume. We exploit thereafter the temporal constraints on the events in order to refine the results of the estimation.
The paper is organized as follows: In section 2, we present the Time Petri Net formalism. A representation of the state of such model is also proposed. Section 3 is dedicated to the discussion of the Enumerative Approach, a reachability analysis method for TPNs, and in the fourth section we present our monitoring approach. Finally, we conclude the paper and we present the prospects of this work in the last section.

2. TIME PETRI NET: PRESENTATION - ANALYSIS

2.1 Definition

Let $\mathbb{T} \subset \mathbb{Q}^+$ be a temporal field, a Time Petri Net (Merlin, 1974) on $\mathbb{T}$ is a 6-tuple $N =< P, T, B, F, M_0, SIM >$ such that:

- $N =< P, T, B, F, M_0 >$ is a marked Petri Net (PN), ($B =$ *backward* and $F =$ *forward*),
- $SIM : T \rightarrow \mathbb{T} \times \mathbb{T}^\infty$ is the Static Interval Mapping, which associates to each transition in T its static firing interval, with rational bounds of firing (as $\mathbb{T} \subset \mathbb{Q}$). We say here *static* firing interval, because when studying the dynamics of the TPN, these intervals will then evolve and one would then speak about

dynamic firing interval.
For $t \in T$ such that $SIM(t) = [\alpha, \beta]$, t can fire only when it remains enabled between α and β $t.u$ (time unit), and must fire if it stays enabled during β $t.u$.

2.2 State of a Time Petri Net

The state of a given TPN can be represented by a pair $E = (M, I)$, where M is the marking of the net, and I is the firing interval mapping which associates to each transition t in T its firing temporal interval. The initial state is defined by $E_0 = (M_0, I_0)$, where M_0 is the initial marking and I_0 is the mapping associating to each transition enabled by M_0 its static firing interval, and the empty interval for all other transitions. Formally I_0 is defined by:

$$I_0(t) = \begin{cases} SIM(t) \ if \ \forall \ p \in P : M(p) \geq B(p,t) \\ \emptyset \ otherwise \end{cases}$$

Since time is continuous, and as each enabled transition can fire at any time during its firing interval, it becomes impossible to enumerate all states of a TPN, as opposed to the case of usual PN with a finite number of markings. In this case the set of states can be represented by the marking graph. In other terms, and according to the firing rules discussed in section 2.2, each state can have an infinity of successor states, except for the particular case where the interval $[0, \infty[$ is associated to each transition of the net. The TPN in this case is in fact equivalent to the usual PN obtained by removing the firing intervals associated to the transitions.
To have an idea on the semantics and the expression capacity of TPNs, the reader can refer to (Diaz and al., 2001).

3. BEHAVIOURAL ANALYSIS - REACHABILITY ANALYSIS BY THE ENUMERATIVE METHOD

3.1 Introduction

In order to obtain a finite representation of the states of a TPN, (Berthomieu and Menasche, 1982) proposed the concept of **State Class**. This concept represents the base of the enumerative method of reachability analysis in TPNs which will be discussed in paragraph 3.4.

3.2 Definitions

As explained in section 2.2, a state of a TPN is a pair $E = (M, I)$, M being the current marking of the net, and I the firing interval mapping. A more convenient representation which replaces I by the **Firing Domain** D (see Definition 1 hereafter) is defined in (Berthomieu and Menasche, 1982), (Diaz and al., 2001):

Definition 1. For a state E, let $T_e = \{t_i / \ M \geq B(t_i)\}$ be the set of enabled transitions. The **Firing Domain** D is the set of vectors solutions of the system of linear inequations in τ_i (possibly parameterized), where τ_i represents the relative firing date of transition t_i.
In the vectors of D, the i^{th} component corresponds to the τ_i variable. Note that the system of linear inequations on τ_i variables for which D is a solution, can be written in the form $A.\underline{t} \leq b$ (Aspvall and Shiloach, 1980), where $\underline{t}$ is the vector whose components correspond to τ_i variables.

Definition 2. A **Dated Firing Sequence (DFS)** (Diaz and al., 2001) is a pair (s, u), where s is a realizable firing sequence of transitions ($s \in T^*$), and u is the sequence of firing dates of the transitions in s. Here s is called the **Firing Support** of the DFS (s, u).

3.3 State class

State classes bring a generalization of the notion of the TPN state. Indeed, instead of considering the state reached by the net from its initial state following the occurrence of a DFS (s, u), a state class represents all states reachable from the considered initial state following the occurrence of all DFSs which have as firing support s.
Thus, a state class is associated to a realizable firing sequence from the initial state.
More explicitly, a state class C associated to a realizable firing sequence s is the pair (M, D), where M is the marking obtained after the firing of s starting from the initial state, and D is the firing domain of all reachable states, starting from the initial state, by the realization of all achievable DFSs having as firing support s. In the rest of this paper, we will use the term 'class' to indicate a state class.

3.4 Reachability search by the enumerative method

In this section, we will discuss the rule of transition between classes, as well as the steps to be followed to obtain the graph of classes of a given TPN.

3.4.1. rules of firing Let $N = < P, T, B, F, M_0, SIM >$ be a TPN, and $t_i \in T$.
t_i is firable starting from a given class $C = (M, D)$ iff:

(1) $M \geq B(t_i)$.
(2) (a) $A.\underline{t} \leq b$.
(b) $\tau_i \leq \tau_j \ \forall \ i \neq j$.

The first condition (1) is the usual firing condition in PNs. Conditions (2.a) and (2.b) translate the fact that the firing domain is respected, and that t_i can be fired first among all enabled transitions.

3.4.2. Transition between classes (Berthomieu and Menasche, 1982), (Diaz and al., 2001)
Consider that starting from a given class $C = (M, D)$, the system state reaches the class $C' = (M', D')$ following the firing of the transition t_i,

(1) The new marking M' is determined as in usual PNs: $M' = M - B(t_i) + F(t_i)$.
(2) The new firing domain D' is determined starting from the linear system $A.\underline{t} \leq b$ of D, according to the following algorithm:
   (a) Conditions (2.b) quoted in 3.4.1 are added to the linear system $A.\underline{t} \leq b$.
   (b) All variables τ_j associated to transitions t_j in conflict with t_i are dismissed from the system.
   (c) Each variable τ_k / $k \neq i$ is replaced by the sum $\tau_i + \tau_k$. Then, τ_i is eliminated from the system.
   (d) For each transition t_l newly enabled (by M'), a new variable τ_l framed by the bounds of the static firing interval of t_l is introduced to the linear system.

4. DEVELOPMENT OF A MONITORING APPROACH BASED ON THE GRAPH OF CLASSES

4.1 Introduction/Problematic

The objective of our approach is to determine the state of the system, starting from some given dated occurrences of observable events. In addition the approach attempts to find the scenarios (of events) which could make the system evolve from a starting state to its current state determined while applying the approach (following the occurrence of the last observable event). Concretely, the approach uses a TPN model of the system, which specifies the normal and abnormal (or failing) behaviors of the system. In this model, each transition corresponds to a different event. Let us denote T_o the set of transitions corresponding to observable events, and T_{un} the set of those corresponding to unobservable events.
The method to obtain the behavioural model of the system is out of the scope of this paper.

4.2 Principle

4.2.1. Introduction This section presents the principle of our monitoring approach as well as the various steps to follow in order to implement this approach. The goal is to determine the state of the system, starting from the timed occurrences of observable events. For that, a tool allowing to *estimate* the system state is devised. Let us call this tool **Estimator** (do not confuse with the notion of Estimator as defined in the continuous-Automatic field).
Moreover, additional methods and algorithms allow refining the result (estimation) provided on the basis of the estimator, by exploiting the temporal constraints related to its states. Indeed, this information is used to decrease the number of possible evolution scenarios in the system starting from its last determined state. We will next discuss the various stages to obtain our Estimator.

4.2.2. Building the graph of classes The steps to follow in order to obtain the graph of classes were explained in paragraph 3.4.2. However, a condition must be checked at this stage. Indeed, in order to be able to work out the graph of classes of our TPN model, the number of classes must be finite. Several research tasks relating to the property analysis of TPN models were undertaken (Diaz and al., 2001), and some theorems were proven. The theorem which interests us most is the following:

Theorem. *The number of classes of a Time Petri Net is finite if and only if this net is bounded.*
Moreover, many other theorems that propose sufficient conditions for the *bounded* property have been developed. These theorems with their proofs can be found in (Diaz and al., 2001).

4.2.3. Elaboration of the Estimator The idea to use an estimator in the evaluation of the system state was inspired from the research works of Lafortune and Sampath (Lafortune and Sampath, 2000), (Sampath *et al.*, 1996) who work out **Diagnosers** for DESs starting from event models. (Ushio *et al.*, 1998) propose also to build observers based on PN models in order to monitor DESs. Here we exploit in addition the temporal information through methods we develop in order to refine the monitoring results.
In fact, our Estimator is a graph similar to the graph of classes, except that the transitions between its nodes are done by transitions which correspond to observable events (transitions in T_o) only. Also the nodes of the Estimator correspond to macro-states that could contain several classes.
In order to be able to build our Estimator, we define a set of mappings on the set $\mathcal{C}$ of classes:

- The transition mapping, which manages the transitions between the classes:
  $f : \mathcal{C} \times T^* \longrightarrow \mathcal{C}$
  Let c_i and c_j be two classes in $\mathcal{C}$, and s a transition sequence in T^*:
  $f((c_i, s)) = c_j$ iff s connects c_i to c_j in the graph of classes. This mapping verifies: $f((x, st)) = f((f((x, s)), t))$
- The mapping of unobservable reachability $\mathcal{UR}$, which enables to find the set of classes reachable, from a given class c, after the firing of all unobservable sequences (sequences of T^*_{un}) which are realizable starting from c.
  $\mathcal{UR} : \mathcal{C} \longrightarrow \mathrm{P}(\mathcal{C})$
  $c_i \longmapsto \mathcal{UR}(c_i) = \{c_j \in \mathcal{C} \ / \ \exists \ s \in T^*_{un}, f((c_i, s)) = c_j\}$
  $\mathrm{P}(\mathcal{C})$ being the set of partitions of $\mathcal{C}$.

We will explain now how to obtain the nodes of the estimator. The initial node N_0 in the Estimator contains the initial class c_0 as well as the set of classes reachable starting from c_0 after the firing of a sequence of T^*_{un}. The set of classes of N_0 can so be represented by: $\{c_0\} \cup \mathcal{UR}(c_0)$.
The set of nodes of the Estimator is obtained by following the algorithm below:

(1) For each class c_i of N_0, find in the graph of classes the set $T^{c_i}_{obs-next}$ of transitions corresponding to observable events (transitions in T_o) which are firable starting from c_i. In the graph of classes, these transitions correspond to the arcs leaving c_i and labeled with a transition of T_o. Let us denote $T_{N_0-obs-next}$ the set formed by reuniting the $T^{c_i}_{obs-next}$ sets relative to all classes c_i in N_0, i.e:
$$T_{N_0-obs-next} = \bigcup_{c_i \in N_0} T^{c_i}_{obs-next}$$
Note that $T_{N_0-obs-next}$ is a set in the mathematical sense of the term. That means an element does not repeat.

(2) For each transition t_i of $T_{N_0-obs-next}$, a new node N_i is created and a directed arc labeled with t_i connects N_0 to the node newly created.

(3) For each transition $t_i \in T_{N_0-obs-next}$, we denote O_i (O=Origin) the set of classes of N_0 which have an arc labeled with t_i, in the graph of classes, connecting them to a given class c_{ij}.
The node N_i created will thus contain all classes c_{ij} as well as all classes reachable, from the c_{ij} classes, following the firing of unobservable sequences (from T^*_{un}). Formally, the set of classes in N_i can be represented as follows:
$$\bigcup_j \{\{c_{ij}\} \cup \mathcal{UR}(c_{ij})\}$$

(4) Reiterate steps (1), (2) and (3) for all classes of each node N_i (instead of N_0) in order to find their, respectively, next nodes, and so on.

4.2.4. Equivalence between Nodes We will introduce now two new concepts which will be used thereafter.
Let N_i and N_j be two nodes of the Estimator, such that there is a directed arc $N_i \longrightarrow N_j$, labeled with a transition t_k (of T_o).

Definition 1. We call **Set of Entry Classes** of N_j (**SEC**(N_j)), the set of all classes of N_j, obtained as result of the firing of t_k starting from a class in N_i.

Definition 2. The **Set of Shadow Classes** of the node N_j (**SSC**(N_j)) corresponds to the set of all classes in N_j, obtained following the firing of all realizable unobservable sequences starting from classes in **SEC**(N_j).
In other terms, for each node N, we have: $SSC(N) = \mathcal{UR}(SEC(N))$.

Definition 3. Let N_i and N_j be two nodes obtained by following steps (1) to (4) of the Estimator building algorithm.
N_i and N_j are **equivalent** ($N_i \Leftrightarrow N_j$) iff $SEC(N_i) = SEC(N_j)$ and $SSC(N_i) = SSC(N_j)$.

Remark 1. The set SEC of the initial node N_0 is the singleton $\{c_0\}$, where c_0 is the initial class in the graph of classes.
Remark 2. The classes set of a given node N can be written as: $SEC(N) \cup SSC(N)$
Remark 3. Sets SEC and SSC of a given node are not inevitably disjoint.
Remark 4. The Set of Entry Classes of a given node N can be obtained starting from any arc entering to N.
Property. *Given N_i and N_j 2 nodes obtained while building the Estimator:*
IF $SEC(N_i) = SEC(N_j)$ THEN $N_i \Leftrightarrow N_j$.
Proof:
$SEC(N_i) = SEC(N_j)$ (1)
According to definition 2, we have:
$SSC(N_i) = \mathcal{UR}(SEC(N_i))$ and
$SSC(N_j) = \mathcal{UR}(SEC(N_j))$
Then from (1) we obtain
$SSC(N_i) = SSC(N_j)$ (2)
Hence, according to definition 3, (1) and (2) imply:
$$N_i \Leftrightarrow N_j$$

Remark 5. While building the estimator by applying the algorithm presented above, a new node N_i, with a set SEC_i of entry classes, is created (steps (2) and (3)) only when, among the nodes already obtained, none has SEC_i as set of entry classes. In other terms, a new node N_i is created only when, among the nodes already obtained, none is equivalent to N_i (Property 4.2.4).

The Estimator nodes can be represented as in figure 1.

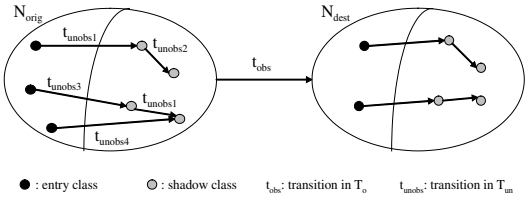

Fig. 1. General form of Estimator nodes

4.3 On-line Monitoring / Use of the Estimator

4.3.1. Introduction Let us recall here that our objective is to monitor (detection/diagnosis) a system for which we know in advance the "complete" factual behavior in time.
The proposed monitoring approach is based on the use of the Estimator at each occurrence of a new observable event. It allows tracking the evolution of the system state. The initial state of the system can be represented by class c_0 of node N_0. When time passes (without occurrence of observable events), the state of our TPN can be represented by c_0 or one of the classes of $SSC(N_0) = \mathcal{UR}(c_0)$. As soon as an observable event e_k (represented by a transition t_k) occurs at a given date θ, the system state moves from the node N_0 to the node N_i obtained by following the outgoing arc of N_0 labeled with t_k. Thus, the Estimator provides the following information: the system state after the occurrence of e_k can be represented by one of the classes in $SEC(N_i)$.
The next paragraph shows how to refine this result.

4.3.2. Enrichment of the Estimator The result we obtain based on the Estimator can be refined by exploiting the temporal data (occurrence dates of observable events, obtained by on-line observation) and temporal constraints on the system behavior (from the TPN model of the system). The refining of these results is done by eliminating the scenarios which do not respect the identified constraints.
Note that the graph of classes does not give the set of realizable Dated Firing Sequences (DFS) for a given firing sequence. Indeed, the relative firing dates in a DFS are not always independent (for example when many transitions not in conflict are simultaneously enabled).
However, there is a relation between the classes firing domains and the date sequences relating to realizable DFSs. Moreover, there is a technique making it possible to obtain a framing for the entire duration of a given realizable firing sequence σ (Diaz and al., 2001). The technique consists in applying the algorithm presented in paragraph 3.4.2 on the classes resulting from the sequence σ, but instead of removing variable τ_i corresponding to the fired transition t_i, in the 3^{rd} step, τ_i will be just replaced by a new variable θ_i. Thus θ_i becomes a parameter of the linear system of the class obtained after the firing of t_i and possibly of the following classes reached as a result of the realization of σ.
Hence, the linear system corresponding to the firing domain of the last obtained class, will contain as many parameters θ_i as transitions in the considered sequence σ. Any solution $\underline{\theta} = (\theta_1, \ldots, \theta_n)$ of this last linear system (Aspvall and Shiloach, 1980) represents a sequence of possible relative dates for a DFS whose firing support is σ. Thanks to this technique, checking of temporal properties, like temporal framing of the firing sequences duration, or establishing the time interval during which the system has a given marking, becomes possible. And in this way, we will be able to add additional information to our Estimator.
Concretely, during the building of the Estimator, each time a new arc labeled t is created from a node N_{orig} towards a node N_{dest}, for each class c_j of $SEC(N_{dest})$:

- find in $SEC(N_{orig})$ all classes c_i which have in the graph of classes, a firing sequence (scenario) st, where $s \in T^*_{un} \cup \emptyset$, which connects it to c_j
- For each pair (c_i, st), determine the minimal duration (d_{min}) and the maximum duration (d_{max}), of the firing of the sequence st starting from c_i. Durations d_{min} and d_{max} correspond in fact to the bounds of the time interval during which the firing of t is expected if the scenario st effectively occurs starting from the c_i class
- For each pair (st, c_i), we dedicate a line in the table associated to the class c_j. The *Origin Class* column corresponds here to the c_i class. The *Scenario* column corresponds to the firing sequence st, and the *Interval* column corresponds to the interval $[d_{min}, d_{max}]$

Each class in the set SEC of a given node in the Estimator will have the following form:

$C_j(M_j, D_j)$		
Scenario	**Origin class**	**Interval**
$s.t$	c_i	$[d_{\min}, d_{\max}]$
...	...	...
...	...	...

Table 1. General form of classes in the SEC set

4.3.3. On-line follow-up We next discuss, the way in which the monitoring activity will be carried out in a dynamic way through the Estimator. Initially, the state of the system is in node N_0 (in c_0 more precisely). Following the occurrence of a given observable event e represented by a transition t, the system state reaches the node N_i obtained while following the outgoing arc of node N_0, labeled with t. More precisely, the new state can be represented by one of the classes in $SEC(N_i)$. Thereafter, the following algorithm is applied:

(1) For each class in $SEC(N_i)$, exclude scenarios other that those in the form st where $s \in T^*_{un} \cup \emptyset$.
(2) Among the scenarios remaining in the classes of $SEC(N_i)$, exclude scenarios for which the *origin class* does not belong to $SEC(N_0)$ and/or the date when e occurred does not belong to the interval corresponding to the scenario.
(3) Exclude the classes of $SEC(N_i)$ for which all associated scenarios were excluded. Let us denote $\widetilde{SEC}(N_i)$ the set of classes in $SEC(N_i)$ which were not excluded.
(4) In $SSC(N_i)$, exclude all classes of the set $(SSC(N_i) - \mathcal{UR}(\widetilde{SEC}(N_i)))$.
(5) Each time a new observable event occurs, reiterate steps (1) to (4) on the reached node but by considering only the remaining (not excluded) classes of the preceding node.

Interpretation. The first step corresponds to the suppression of the scenarios for which origin class does not belong to the node N_0. Step (2) eliminates the scenarios which do not respect the temporal constraints on the event as they were expressed in the system model (TPN). In the 3^{rd} step, we eliminate classes whose no scenario is effectively realizable knowing the temporal data and constraints. One keeps in step (4) only the classes which will be reached starting from the remaining classes. Finally at the 5^{th} step, one repeats in the same way steps (1) to (4) on the new node (instead of N_0) reached following the occurrence of an observable event.
Hence, the result of the monitoring analysis process after the occurrence of e can be represented by the remaining scenarios.

5. CONCLUSION

The monitoring approach we developed in this paper is characterized by the use of temporal constraints to refine the monitoring process result obtained by using only observable events. It is about an on-line monitoring approach which requires an advance knowledge of the system behavior.

In future work, we will try to improve our approach by integrating it with the technique of Causal Temporal Signatures (CTS) (Toguyéni *et al.*, 1997) developed in our laboratory, and whose general idea is to exploit the occurrence of the events progressively to decrease the number of former scenarios which were considered to be realizable. The interpretation of some missing of observable events until fixed temporal terminal will also be taken into account in the framework of our approach. In addition, we will extend the TINA tool (Roux and Berthomieu, 1986) which implements the enumerative method by integrating our monitoring approach. That will allow automatic building of the Estimator and automatic on-line follow-up of the system state.

REFERENCES

Aspvall, B. and Y. Shiloach (1980). A polynomial time algorithm for solving systems of linear inequalities with two variables per inequality. *SIAM Journal on computing* **9**, 827–845.

Berthomieu, B. and M. Menasche (1982). A state enumeration approach for analysing time petri nets. *Proceedings of applications and theory of Petri nets (ATPN'82)* pp. 27–56.

Diaz, M. and al. (2001). *Les réseaux de Petri, modèles fondamentaux*. Hermès. Paris.

Lafortune, S. and M. Sampath (2000). Discrete event systems approach to failure diagnosis: Theory and applications. *11ème International Workshop on Principles of Diagnosis*.

Merlin, P. (1974). A study of the recoverability of computer system. *PhD thesis, university of Californie*.

Roux, J.L. and B. Berthomieu (1986). Verification of a local area network protocol with tina: A software package for petri nets. *Proceedings of the 7 workshop on applications and theory of Petri nets* pp. 183–205.

Sampath, R., K. Sinnamohideen, S. Lafortune and D. Teneketzis (1996). Failure diagnosis using discrete event models. *IEEE Transactions on Systems Technology* **4**, 105–124.

Toguyéni, A., E. Craye and J.C. Gentina (1997). Time and reasoning for on-line diagnosis of failures in flexible manufacturing systems. *Proceeding of 15th IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics, Berlin, Germany* **6**, 709–714.

Ushio, T., I. Onishi and K. Okuda (1998). Fault detection based on petri net models with faulty behaviors. *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics* pp. 113–118.