Monitoring of timed discrete events systems: Application to manufacturing systems

Adib Allahham, Hassane Alla

Monitoring of timed discrete events systems: Application to manufacturing systems

Adib Allahham, Hassane Alla

Laboratoire d'Automatique de Grenoble/ENSIEG-INPG
Rue de la Houille Blanche - Domaine universitaire BP46 , Saint Martin d'Hères , 38402, France
{Adib.al-lahham,hassane.alla}@inpg.fr
www.lag.ensieg.inpg.fr

Abstract— **In this paper, we develop a method for fault detection based on Stopwatch automata. This method takes into account the exact behavior of some physical resources. Because of a malfunction, some system tasks can be stopped and resumed a little later. Thus, we introduce the concept of an acceptable system behavior which is supposed to be observable. It is supervised by two clocks for each task. The timed state space of the stopwatch automaton delimits exactly the acceptable system behavior. It permits to detect the fault as early as possible. This space is a set of the inequality constraints where any violation of these constraints represents a fault.**

I. Introduction

Fault detection in the complex systems plays an important role for economic reasons and for security and reliability motives.
Solving detection problems for complex systems is a complicated task requiring a systematic method. As a result, this problem has been received a considerable attention in the literature. One of the used detection methods is the model-based method. The general principle of all model-based approaches is to compare the expected behavior of the system, given by a model, with its actual behavior. Two kinds of models may be used, depending on the objectives of the monitoring system. The normal behavior model represents the system in normal situations, when no fault is present. In this case, the model contains the timing and sequencing relationships of the events in normal behavior. Typically, these models are based either on Timed Petri nets or on timed automata. Procedures based on the normal behavior model are a priori able only to detect faults. The watchdog mechanism is used in this kind of procedure [12], [13] and [10]. The faulty behavior model represents the system in faulty situations, taking explicitly into account the influence of the faults [4]. It is also possible to describe the complete system behavior which includes the normal and faulty behaviors of the system at a time. This model is used when the objectives are to diagnose the faults.The faults are modeled as non observable events [6] and [14].
As we saw, the model represents either the normal behavior or faulty behavior or both. But, these state representations do not include all possible behaviors that can happen during the working of a system as we will see in the following section.

II. problem definition

In this section we describe the behavior of some real systems. Between a normal state and a faulty one, an intermediate state can appear. In this state, the system can come back to the normal behavior or it leaves toward a faulty state (Fig.1).
In this paper, we treat the fault detection in the complex

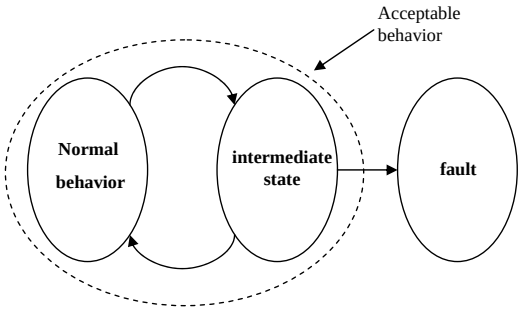

Fig. 1. The considered system behavior

systems where their physical components can be subjected to unpredictable malfunctions during the execution of their tasks. This observable malfunction causes a task interruption and leads the task to it intermediate state. The task subjected to this malfunction is called preemptive task and its behavior including normal and intermediate states is called the acceptable behavior.

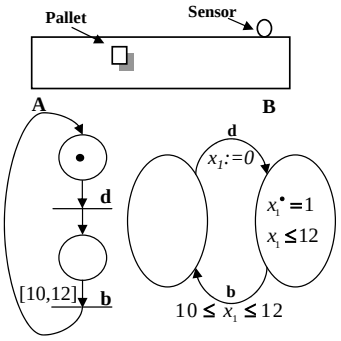

Fig. 2. The conveyor and existing monitoring methods

To explain this behavior, we treat the system given in Fig.

2. A conveyor transfers a pallet from point A to B. The conveyor begins its task when the control system gives the order d. The task end is represented by the event b. The transported pallet moves at a constant speed. The pallet needs 10 $t.u$ in order to move from A to B. The acceptable duration to execute this task is $[10, 12]$ $t.u$ taking into account the transitory stops. In Fig. 2, we show the proposed models in the literature for monitoring this system. Consider now that this conveyor is a component of a complex system. The existing monitoring techniques consider that the reachable timed space is a hypercube (Fig. 3). Let us suppose that the pallet is blocked in a permanent manner at point A. For the models in Fig. 2, the fault is detected when the time exceeds the instant 12. We propose a more precise approach to detect the faults as early as possible. This requires to calculate the exact reachable space.

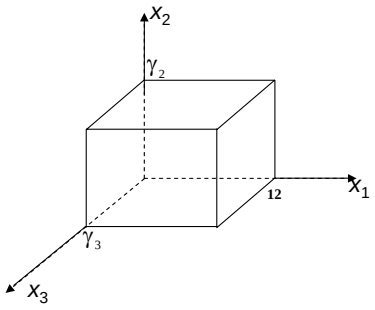

Fig. 3. The timed reachable state of the models given in the literature

The rest of this paper is organized as follows: Section 3 describes our solution intuitively using the stopwatch automata which we define in Section 4. Section 5 presents our method in a formal manner. Section 6 precise the detection mechanism of our monitoring system. We treat an illustrative example in Section 7. Finally, Section 8 concludes the paper with a summary and the future works.

III. Intuitive presentation of the approach

To show our approach, we construct the monitoring model of the example given in Fig. 2.
As soon as the control system gives the order d, we initialize two clocks: x and t (Fig. 4). The dynamic of x reflects the task state.

- $\dot{x} = 1$ the conveyor transfers the pallet.
- $\dot{x} = 0$ the pallet is stopped on the conveyor.

Then, the clock x represents an indicator on the position of the pallet on the conveyor.The clock t measures the time passed since the task has been started.

- $\dot{t} = 1$ until the occurrence of event b.

When event b occurs, the value of t indicates if the task has been achieved in the acceptable interval $[10, 12]t.u$.

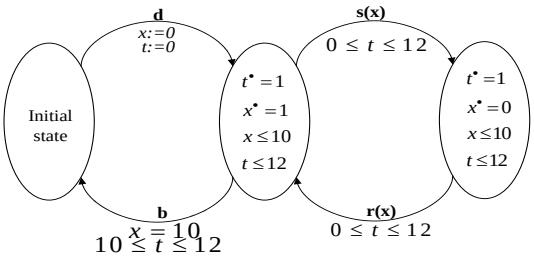

Fig. 4. The proposed monitoring model of the conveyor

Normally, the pallet moves on the conveyor and arrives at its end $x = 10$ at the instant $t = 10$. During this task, the pallet can be blocked temporarily or in a permanent manner. Blocking of the pallet (event $s(x)$) involves a change of the system state. In this new state, the clock x is freezed because the pallet does not move on the conveyor while t remains active. The condition associated to this change is $0 \leq t \leq 12$ since the blocking can happen at any position on the conveyor before exceeding the acceptable interval. When the transfer task resumes (event $r(x)$ arrives), the system come back to it previous state and the clock x starts again. The guard associated with this transition is $0 \leq t \leq 12$ because the task must be executed at any time during the acceptable interval.
The values $x = 10$ and $10 \leq t \leq 12$ expresses that the pallet has arrived to point B, in the acceptable interval. Our objective is to determine all the possible trajectories that permit to reach these values. It means constructing the timed state space permitting to arrive to these values. Fig. 5 shows this state space (the parallelogram).
To show the utilization of this space, we treat the following cases given in Fig. 5:

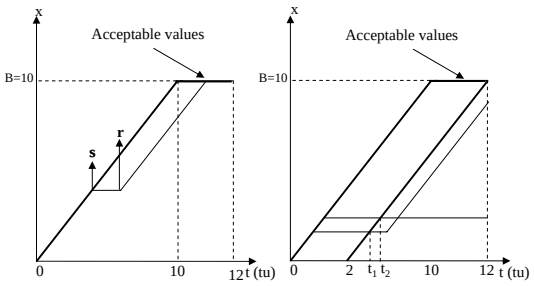

Fig. 5. The new timed state space of the conveyor

- The monitoring system triggers an alarm at instant t_1 even though the pallet has not been blocked definitely, because whatever it happens, the transfer duration will be greater than 12 $t.u$.

- The pallet has been blocked definitely; our system triggers an alarm at instant t_2. It does not wait until 12 $t.u$ to trigger an alarm.

We notice that our monitoring model is a stopwatch automaton. It is characterized by its capacity to modeling

the preemption while the models given in Fig. 2 do not permit to represent this behavior. They do not keep the information at the instant and position at which the task has been interrupted to continue the working. There are other tools for modeling this behavior as Preemptive Time Petri Nets (PTPN) [3], but we chose the stopwatch automata thanks to the general characteristic of timed automata to represent the exact timed state space [8].

IV. Stopwatch Automata

We basically define stopwatch automata (SWA) as timed automata in which the clocks can be suspended then resumed later [11]. It is also a class of linear hybrid automaton (LHA) where the derivative of a variable in a location can be either 0 or 1 [1].

Definition. 1: A Stopwatch automaton is a 7-tuple $(L, l_0, X, \Sigma, A, Inv, Dif)$ where
- L is a finite set of locations,
- l_0 is the initial location,
- X is a finite set of positive real-valued clocks,
- Σ is a finite set of actions,
- $A \subset L \times C(X) \times \sigma \times 2^X \times L$ is a finite set of edges. $a = (l, \delta, \sigma, R, l') \in A$ is the edge between the locations l and l', with the guard δ, the action σ and the set of clocks to reset R. $C(X)$ is the set of constraints over X.
- $Inv \in C(X)^L$ maps an invariant to each location,
- $Dif \in (\{0,1\}^X)^L$ maps an activity to each location, $\dot{X}$ being the set of derivatives of the clocks w.r.t time.$\dot{X} = Dif(l)(x)_{x \in X}$. For short, given a location l and a clock x, we will denote $Dif(l)(x)_{x \in X} = \{0, 1\}$.

□

A state of the SWA is a pair (L, E) where L is a location of SWA and E is its timed state space.
We present some techniques that we will use in state space analysis.
When a system reaches a location L_n, the active clocks have several possible valuation. The set of these valuations defines the timed space at the entry of L_n : E_n^a [5].
An automaton have two possibilities of evolution from a state L_n:
- remaining in the same location while the time progresses. The reachable state following this evolution is calculated by the continuous successor Suc_t. Therefore, the E_n in the state L_n is:

$$E_n = Suc_t(E_n^a) \tag{1}$$

- firing a transition $a = (L_n, g_{n,n+1}, \sigma, R, L_{n+1}) \in A$. The timed reachable state following this evolution is calculated by the discrete successor Suc_d.

$$Suc_d(E_n) = E_n \wedge g_{n,n+1} \wedge R\{x := 0\} \tag{2}$$

These analysis methods are called forward analysis methods. We also define one of the backward analysis methods, the only one which is useful in our work. It is the continuous predecessor of a space Pre_t. It is the space E from which the system can reach a given space Q by letting the time progress and in staying in the same location L_n.

$$E = Pre_t(Q). \tag{3}$$

V. System model

We consider the monitoring problem of the complex system S. Let $\Im$ the set of tasks of S and T_{pre} the set of the preemptive tasks such that $T_{pre} \subseteq \Im$. In order to model the behavior of S, we begin with modeling a preemptive task $T_i \in T_{pre}$, then we describe the global behavior of S.

A. Behavior of a preemptive task

The considered task T_i has a known execution duration $[\alpha_i, \beta_i]$. Because of the interruptions and for the productivity motive, the acceptable duration for execution T_i is also known $[\alpha_i, \gamma_i]$ where $\beta_i < \gamma_i$.

Hypothesis. The execution speed of the task T_i is constant or it varies around a mean value. This speed variation is taken into account by the interval $[\alpha_i, \beta_i]$.

□

This type of tasks represents most services executed by the resources in manufacturing systems (processing, transport, filling, .etc.). So, this hypothesis is realistic and necessary in our modeling point of view. Considering the properties of the tasks mentioned over, we distinguish the following behavior of a preemptive task given in Fig. 6:

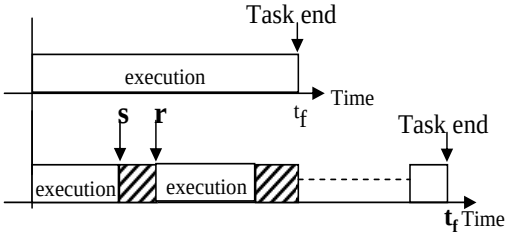

Fig. 6. The behavior of a preemptive task

- T_i is executed without interruption; therefore the execution duration belongs to the interval $t_f \in [\alpha_i, \beta_i]$.
- T_i has been executed but with several interruptions. It resumes a little letter from the position at which it has been interrupted. In this case: $t_f \in [\alpha_i, \gamma_i]$.

B. Modeling of a preemptive task

To model the behavior of T_i, we need to use a Stopwatch automaton with two states such that the automaton can move back and forth between "normal execution" and "preempt" as in Fig. 7.

Definition. 2: Let $(\Gamma, L_1, X, \Sigma_i, A, Inv, Dif)$ the Stopwatch automaton of task T_i where
- $\Gamma = \{L_1, L_1'\}$,
- L_1 is the initial location,

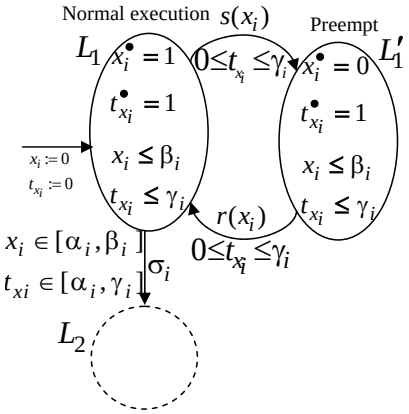

Fig. 7. The stopwatch automaton of a preemptive task

- $X = \{x_i, t_{x_i}\}$,
- $\Sigma_i = \{\sigma_i, s(x_i), r(x_i)\}$,
- $A = \{a_1, a_2, a_3\}$,
- $Inv(L_1) = Inv(L'_1) = \{x_i \le \beta_i, t_{x_i} \le \gamma_i\}$;
- $Dif(L_1)(t_{x_i}) = Dif(L'_1)(t_{x_i}) = 1$,

$Dif(L_1)(x_i) = 1, Dif(L'_1)(x_i) = 0$.

□

The events $s(x_i)$ and $r(x_i)$ represent the stop and the resumption of task T_i respectively while event σ_i defines the end of this task.
The clock x_i adds up the durations of partial execution of task T_i while t_{x_i} measures the total execution duration.
In this automaton, we find the following transitions.

- $a_1 : L_1 \xrightarrow{s(x_i)} L'_1$, represents the interruption of T_i. This stop can happen at any instant during the execution of the task, before exceeding the acceptable duration, therefore the guard of this transition is $g_1 = 0 \le t_{x_i} \le \gamma_i$;
- $a_2 : L'_1 \xrightarrow{r(x_i)} L_1$, represents the resumption of T_i. This must happen before exceeding the acceptable duration, therefore the guard of this transition $g_2 = g_1$;
- $a_3 : L_1 \xrightarrow{\sigma_i} L_2$, represents the execution of T_i, where L_2 represent the initial state of the task T_j. The guard of this transition is $g_3 = \alpha_i \le x_i \le \beta_i \wedge \alpha_i \le t_{x_i} \le \gamma_i$. It represents the task execution in its acceptable duration.

C. State space analysis

- Forward analysis

Generally, when a task T_i starts, clocks x_i and t_{x_i} are initialized. Therefore, the timed state space at the entry of L_1 is $E^a_t = \{x_i = t_{x_i} = 0\}$. The evolution of the automaton from (L_1, E^a_t) is determined by applying a decidable algorithm based on the methods of analysis mentioned in Section 4. It allows to calculate the timed spaces E_t and E'_t in L_1 and L'_1 respectively. The generic steps of this algorithm are:

{
- Reachability analysis in location L_1

$E_t = Suc_t(E^a_t)$
- Reachability analysis in location L'_1

$S_{L_1,L'_1} = Suc_d(E_t)$

$E'_t = Suc_t(S_{L_1,L'_1})$
- Updating the space E_t

$E^a_t = Suc_d(E'_t)$

$E_t = Suc_t(E^a_t)$

}

It result the space E_t shown in Fig. 8.*a*. This space E_t memorizes the set of reachable valuations of x_i and t_{x_i} in L_1 and L'_1. It means, all possible trajectories in these locations. Some of these trajectories are acceptable because they permit to verify the guard g_3, but other not. Our objective is to characterize the reachable space that allows to execute the task. It is the objective of the next paragraph.

- Backward analysis

The valuations of the clocks that satisfy the acceptable execution of task T_i given by g_3 define a region. It is called the acceptable region R^c_i.

$$R^c_i = \alpha_i \le x_i \le \beta_i \wedge \alpha_i \le t_{x_i} \le \gamma_i. \quad (4)$$

We calculate the desired space E_d allowing to reach the space R^c_i by using the backward analysis (Fig 8.*b*). The space characterizing the possible evolution permitting to execute task T_i according to the guard g_3 is given by the intersection of E_t and E_d (Fig. 8.*c*).

$$E_1 = E'_1 = E_t \cap E_d. \quad (5)$$

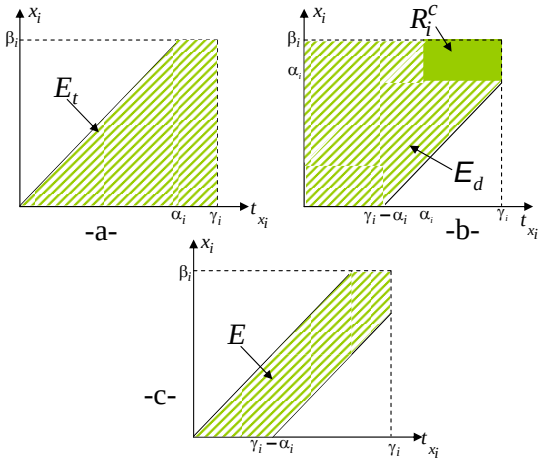

Fig. 8. The timed state space of a preemptive task: (a) reachable, (b) desired, (c) exact

VI. Monitoring System

In Figure 9, we present the monitoring system for a controlled system. The process is composed of a set of tasks $\Im$ dynamically coupled. Let m be the number of the preemptive tasks in $\Im$ where $T_{pre} \subseteq \Im$. The monitoring model

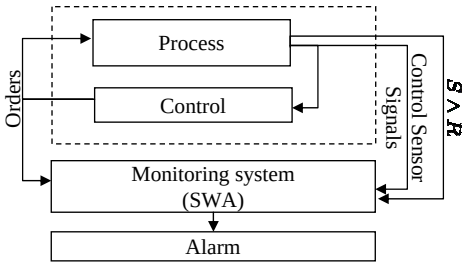

Fig. 9. The entries\output of monitoring system

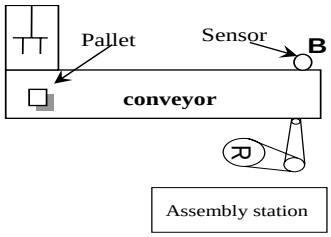

Fig. 11. The manufacturing system

of this system is a stopwatch automaton specified by Definitions 1. It represents the acceptable behavior of the controlled system. The behavior of each preemptive task is given by the model specified in Definition 2. In this monitoring model:

- L is the set of reachable states in acceptable behavior of controlled system. In these states the active and interrupted tasks are taken into account,
- $\Sigma = \Sigma' \cup S \cup R$ where : Σ' represents the events coming from the process and from the controller. $S = \{s(x_1), s(x_2), s(x_3), ..., s(x_m)\}$ and $R = \{r(x_1), r(x_2), r(x_3), ..., r(x_m)\}$ are the sets of stop and resume events coming directly from the process.

The states of monitoring system are updated in permanence by the set of events Σ. We associate to each location a timed space corresponding to the clocks evolution in the acceptable behavior (Fig. 10). This space is calculated based on the techniques explained in Section IV, by using the existing programs in the litterateur such Hytech [9] and Phaver [7]. It represents an algebraic formulation of the dynamic behavior of the system. We have calculated this space for an isolated task T_i in Section V. In fact, this

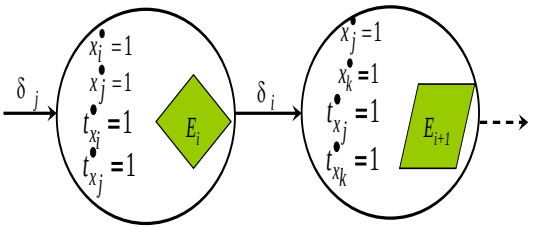

Fig. 10. The proposed monitoring model

space is a set of algebraic inequalities. Consequently, any violation of these inequalities represents a fault which our monitoring system detects immediately.

VII. Illustrative Example

To illustrate the approach that we develop, we consider the following example. A part of manufacturing system is made up of a transfer station and a robot. The transfer station is composed of an actuator and a conveyor. This system est shown in Fig. 11.

When the control system gives the order d, the actuator puts down a pallet on the conveyor. When the sensor B detects the transferred pallet (*event b*), the transfer station comes back to its initial state, and the robot transfers the pallet to the assembly station, if it is not busy. When the robot finishes its task (*event R*), it comes back to its initial state.

The transfer duration of the pallet on the conveyor is $[3, 4]$ $t.u$ from the instant of giving the order d and that of the robot's task is $[2, 3]$ $t.u$. The acceptable durations to execute these tasks are $[3, 5]$ $t.u$ for the conveyor's task and $[2, 4]t.u$ for the robot's task.

The conveyor's task is supervised by the clocks x_2 and t_{x_2} while that of the robot by the clocks x_4 and t_{x_4}. The stop and the resumption are observable by the events $s(x_2)$ and $r(x_2)$ for the conveyor's task and by the events $s(x_4), r(x_4)$ for the robot's task. For simplicity reasons, we give a part of the automata representing the behavior of this system and the corresponding timed spaces in Fig. 12. The timed

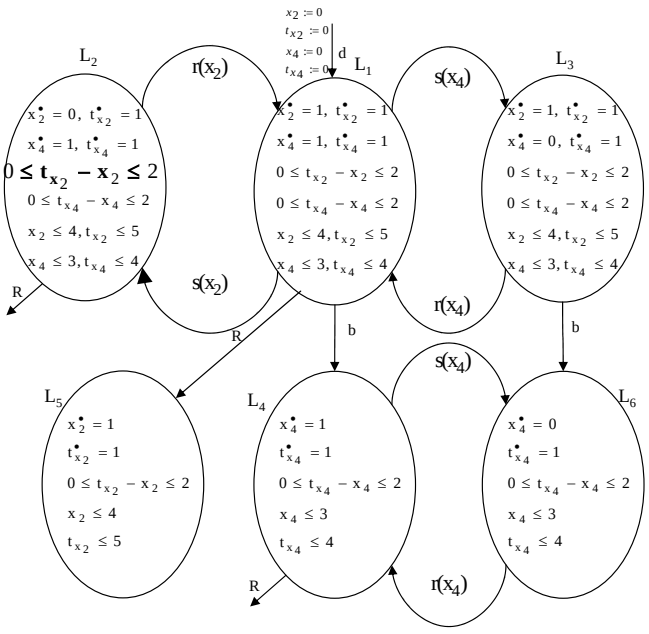

Fig. 12. A part of monitoring model

spaces in the locations L_1, L_2, L_3, L_4, L_5 and L_6 are represented by the algebraic inequalities.

The inequality in bold in L_2 detects the fault presented in Fig. 13 at instant $t_{x_2}(\theta) = 3 + \varepsilon$ where ε is an amount of time infinity small. The corresponding value of x_2 is $x_2(\theta) = 1$. We can explain the setting on the alarm by following reason:

To finish the task supervised by x_2 correctly, one needs to have at least the duration $\alpha_2 - x_2(\theta) = 3 - 1 = 2$ $t.u$. In this case, the corresponding value of t_{x_2} is $t_{x_2} = t_{x_2}(\theta) + (\alpha_2 - x_2(\theta)) = 3 + \varepsilon + 2 > 5$. This execution duration will exceed γ_2 the maximum permitted duration of conveyor's task. So, one detects this fault as early as possible at instant $3 + \varepsilon$.

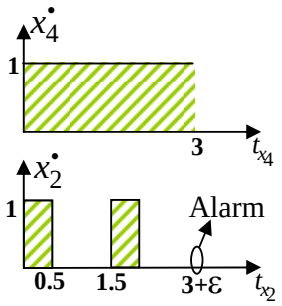

Fig. 13. A faulty scenario.

VIII. Conclusion

In this paper, we have proposed a method for the real-time monitoring systems, based on a stopwatch automata. It takes into account the acceptable behavior of some tasks called 'preemptive tasks'. The locations of this automaton represent the reachable state. These states are updated in permanence by the set of events coming either from the controller or from the process. The acceptable behavior of each preemptive task is supervised by using two clocks. The timed state space in each location of this automaton represents the temporal constraints that the system must respect. These constraints are a set of algebraic inequalities. The monitoring system detects any violation of these constraints. Consequently, it detects the system faults as early as possible. The future works will take into account the faults resulting from a more complex modification of the dynamic of the task that will lead to multi mode behavior (instead of two). Indeed, one of our future investigation include to extend the DBM's data structures [2] to extended DBM (EDBM). This will encode exactly the region of our SWA. So, we will show that due to particular properties of our monitoring model, the SWA is decidable. It has the following properties: (1) valuation of clocks are always bounded with constants, (2) for each task T_i, x_i and t_{x_i} are initialed simultaneously.

References

[1] F. Cassez and K. Larsen, "The impressive power of stopwatch," *11th International Conference on Concurrency Theory (CONCUR'2000)*, Number 1877 in Lecture notes in Computer Science, University Park, USA, Springer-Verlag, pp. 138–152, 2000.

[2] J. Bengtsson and W. Yi, *Timed Automata: Semantics, Algorithms, Tools,* Lectures on Concurrency and Petri Nets, Springer-Verlag,vol. LNCS 3098, 2004.

[3] G. Bucci, A. Fedeli, L. Sassoli and E. Vicario, "Timed State Space Analysis of Real-Time Preemptive Systems," *IEEE Transactions on Software engineering*, vol. 30, No. 2, 2004.

[4] E.D. Dorothy, "An intrusion-detection model," *IEEE Transaction on Software engineering*, vol. 32, pp. 222–232, 1987.

[5] T. Dang, *Vérification et Synthése des systémes hybrides,* PhD thesis, Institut d'Informatique et Mathématique Appliquées de Grenoble, Laboratoire VERIMAG, France, 2000.

[6] M. Ghazel and A. Toguèni and M. Bigang, "A monitoring approach for discrete events systems based on a timed perti net model," *Proceedings of 16th IFAC World Congress* , Prague, 2005.

[7] G. Frehse, " PHAVer: Algorithmic Verification of Hybrid Systems past HyTech," *Proceedings of the Fifth International Workshop on Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 2289, Springer-Verlag, pp. 258–273, 2005.

[8] G.Gardey and O.H.Roux and O.H. Roux, "State space computation and analysis of time Petri nets," *Theory and Practice of Logic Programming*, Special Issue on Specification Analysis and Verification of Reactive Systems, 2006.

[9] T.A. Henzinger, P.-H. Ho and H. Wong-Toi, " HyTech: A Model Checker for Hybrid Systems," *Software Tools for Technology Transfer*, vol. 1, pp. 110–122, 1997.

[10] H. Rayhane, *Surveillance des systèmes de production automatisès : Détection et diagnostic,* PhD thesis, Laboratoire d'Automatique de Grenoble - Institut nNational de Polytechnique de Grenoble (INPG), France, 2004.

[11] O.H. Roux, *Vérification des réseaux de Petri temporels et à chronométres,* HDR thesis, IRCCyN, University of Nantes, France, 2005.

[12] A. Sahraoui and H. Atabakhche and A. Courvoisrier and R. Valette, "Joining Petri nets and knowledge based systems for monitoring purposes," *IEEE, International conference robotics*, pp. 1160–1165, 1987.

[13] V.S. Srinivasan and M.A. Jafari, "Fault detection/monitoring using time Petri nets," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, pp. 1155–1162, 1993.

[14] S. Tripakis, "Fault diagnosis for timed automata," *Proceeding 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, vol. 2791 of Lecture Notes in Computer Science, pp. 205–224, Springer, 2002.