Timed pattern diagnosis in timed workflows: a model checking approach

Yannick Pencolé [*] **Audine Subias.** [*]

[*] *LAAS-CNRS, Université de Toulouse, CNRS,INSA, Toulouse, France (ypencole@laas.fr,subias@laas.fr).*

Abstract
In this article we introduce the problem of timed pattern diagnosis in timed system and propose a method to solve a sub-class of theses problems: the problem of timed pattern diagnosis of workflows. It consists in searching for all the possible evolutions of the workflow that are consistent with a timed sequence of observations and determining whether these evolutions match a pattern of timed events instead of a single fault event. The formal characterization of the diagnosis problem is based on the notion of *pattern matching*. Defined as a reachability problem, the diagnosis problem is then solved by *model checking*.

Keywords: Diagnosis, Time Petri Net, timed pattern, Model checking

1. INTRODUCTION

The problem of fault diagnosis in DESs was introduced in Sampath et al. (1995). Later, extensions of the problem handling time have been proposed using timed automata (Tripakis (2002)) and more recently Petri nets with several interpretations of time and methods, for instance: Chatain and Jard (2005), Jiroveanu and Boel (2006), Basile et al. (2007), Bonhomme (2015), Boel and Jiroveanu (2013), Basile et al. (2015), Liu (2014), Wang et al. (2015). These contributions describe some methods for the same problem that is the diagnosis of single faults in a timed system. In this article we propose to extend the diagnosis problem to more complex behaviors than single events called *timed patterns* and to apply it to timed workflows. *Timed patterns* extend with time the classical notion of pattern (Jéron et al. (2006); Gougam et al. (2017)). Beyond faulty behaviors, classical patterns can represent any untimed normal behavior of interest or any behavior associated with a system specification as in Jiang and Kumar (2004). In Bozzano et al. (2015), such patterns are called *diagnosis conditions* and are represented with LTL (Linear Time Logic) properties with past operators.

As classical patterns, the proposed timed pattern models more complex behaviors than a single event (sequences, concurrent events,...) but it also add the possibility to model a time interval between the occurrence of two events in the patterns. Timed pattern can then naturally model unexpected/unacceptable/faulty time delays between two unobservable events. In this article we firstly propose to formally define the general problem of timed pattern diagnosis in timed system. We then propose a method based on model-checking techniques that solves a subclass of this new problem that is the timed pattern diagnosis in timed workflows. Given the observable part of the run of a workflow, the diagnosis problem consists in determining whether the considered timed pattern has occurred or not during the run of the workflow. The proposed method first converts the problem into a reachability problem and then use model checking tool to solve it. Section 2 formally defines the general problem. The proposed method is presented in Section 3. An illustrative example about a service workflow is described in Section 4.

2. GENERAL PROBLEM STATEMENT

The objective of the approach is to diagnose the occurrence of a timed behavior (i.e a timed pattern) in a timed system. It supposes the implementation of a diagnosis function that takes as input a sequence of observations and returns one of the following three results:

(1) *The pattern's behavior has certainly occurred.*
(2) *The pattern's behavior is certainly absent.*
(3) *The pattern's behavior may have occurred, ambiguity.*

This section defines the different parts of the problem that is: how to model timed systems/timed patterns, how to formally define the occurrence of a timed pattern, what is finally the diagnoser function.

2.1 Preliminaries

This paper deals with timed sequences. Let Σ be an alphabet that is a finite set of symbols, each symbol represents an event. A timed sequence over Σ is an element of $\mathcal{T}(\Sigma) = (\mathbb{N}^+ \times (\Sigma \cup \lambda))^+$. Suppose for instance that $\Sigma = \{a, b, c\}$, $\rho = (a, 3), (\lambda, 1), (c, 1), (c, 1), (b, 2), (\lambda, 5)$ also denoted $3a1\lambda1c1c2b5\lambda$ is a timed sequence, $\rho' = 1b1b40\lambda$ is another one. Symbol λ defines a timepoint (basically $1\lambda2\lambda$ is the same as 3λ). The empty sequence of $\mathcal{T}(\Sigma)$ is given by 0λ.

For any timed sequence it is possible to define a canonical form $\delta_1 e_1 .. \delta_n e_n \delta_{n+1}\lambda, e_i \in \Sigma$ that is obtained by removing useless time points. For instance the canonical form of $\rho = 3a1\lambda1c1c2b5\lambda$ is $3a2c1c2b5\lambda$. In this article and without loss of generality only canonical forms are considered. The projection of a timed sequence ρ over a sub-set Σ_p of Σ (denoted $P^t_{\Sigma_p}(\rho)$) is the timed sequence obtained by removing any symbols that does not belong to Σ_p and by summing the durations between two symbols that are still present after the removal. Let us consider the canonical form of ρ, it corresponds to an event a that occurs at date 3 after the origin of time, an event c that occurs 2 units of time after a (so c occurs at date 5), followed by an event b, 1 unit of time later (so at date 6), and so on. Given $\Sigma_p = \{a, b\}$ then $P^t_{\Sigma_p}(\rho) = 3a5b5\lambda$. Any sub-set (canonical sub-set) of $\mathcal{T}(\Sigma)$ is a timed language over Σ. Let $time(\rho)$ denote the sum of all durations in ρ, then $time(3a4c2\lambda1b2b4\lambda) = 3+4+2+1+2+4 = 14$. ρ' is a prefix of ρ if $time(\rho') \leq time(\rho)$ and there exists ρ'' such that $\rho'\rho''$ and ρ have the same canonical form.

2.2 Matching

Like untimed discrete event system, a timed discrete event system generates a finite set of events Σ. The difference is that in the timed case, the possible time durations between the occurrence of two successive events are defined, whereas in the untimed case, the events are only connected by a precedence relationship. Of course in both cases (timed and untimed), two events may not be related i.e even the order of their occurrence is not known. An evolution of a timed discrete event system is a *timed sequence* on its alphabet. The set of possible evolutions of a timed system is thus a set of timed sequences, that is, a timed language.

Definition 2.1. (Timed system). A timed discrete event system whose set of events is Σ, is a timed language $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ such that if $\rho \in \mathcal{L}$ then any of its prefixes is in $\mathcal{L}$.

The set of events Σ generated by the system is partitioned into two subsets: observable events Σ_o and unobservable events Σ_u. Two timed sequences can be linked by the notion of *matching* that states whether a sequence is *present* inside another.

Definition 2.2. (sequence *matching*). A sequence $\rho \in \mathcal{T}(\Sigma)$ matches a sequence $\sigma \in \mathcal{T}(\Sigma)$, denoted $\rho \Supset \sigma$, if:

- $\sigma = \delta\lambda$ and $time(\rho) \geq \delta$, with: $\delta \in \mathbb{N}$; or
- $\sigma = \delta_0 e_0 \sigma_1$, such that: $\delta_0 \in \mathbb{N}$, $e_0 \in \Sigma$, $\sigma_1 \in \mathcal{T}(\Sigma)$ and there exist two timed sequences $\rho_0, \rho_1 \in \mathcal{T}(\Sigma)$ and a duration $\delta \in \mathbb{N}$ such that:
  (1) $\rho = \rho_0 \delta e_0 \rho_1$;
  (2) $\delta_0 = time(\rho_0) + \delta$;
  (3) $\rho_1 \Supset \sigma_1$.

In other words, ρ matches σ if σ is a sub-word of ρ. Figure 1 illustrates how the sequence $\rho = 2b3c2a5a1c1c3\lambda$ matches the sequence $\sigma = 7a7c0\lambda$. The first event of σ is a and occurs at date 7. In ρ, there is also an event a that occurs at date $7 = 2+3+2$. The second event of σ is c that occurs 7 time units later than a. This event is matched with the third event c of ρ that occurs $5+1+1 = 7$ time units after a. Finally, σ ends instantaneously after event c while ρ lasts 3 time units more so ρ matches σ.

We define a *timed pattern* as a set $\mathcal{R}$ of timed sequences which define a behavior of interest. For example, it may be the occurrence of a single event in a given time period, the occurrence of several events that occur in any order, but whose occurrence dates have constraints or the occurrence of several events that occur in a fixed order with specific time constraints. As in the original diagnosis problem that considers unobservable fault event, we consider that the behavior of the pattern is not observable i.e no event of the pattern is observable in the system.

Definition 2.3. (Timed pattern). A timed pattern over a set of events $\Sigma_{\mathcal{R}} \subseteq \Sigma_u$, is a timed language $\mathcal{R} \subseteq \mathcal{T}(\Sigma_{\mathcal{R}})$ such that if

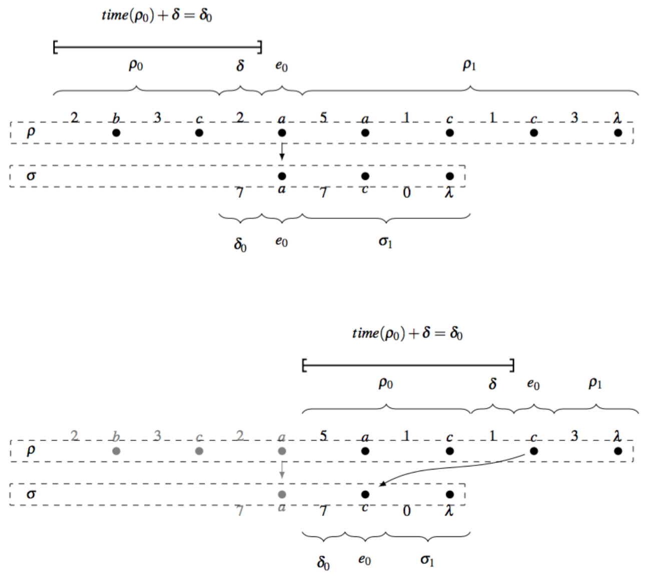

Figure 1. How $\rho = 2b3c2a5a1c1c3\lambda$matches $\sigma = 7a7c0\lambda$: $\rho \Supset \sigma$.

$\rho \in \mathcal{R}$ then only the continuations of ρ of the form $\rho\delta\lambda, \delta \in \mathbb{N}$ are in $\mathcal{R}$.

Note that the timed language of a pattern gives only the required behavior and does not take all the possible events of the system into account. For instance, the pattern *an event a occurs after an event b* is defined by the timed language $\{\delta_0 b \delta_1 a \delta_2 \lambda, \delta_0, \delta_1, \delta_2 \in \mathbb{N}\}$. In this way, the pattern representation is rather in intention than in extension as the representation proposed by Jéron et al. (2006) in the case of untimed patterns that depends on a given set of events. In addition to provide a more concise way to model the pattern, this representation reinforces the separation between the system and the diagnosis objectives. Moreover such a representation is well suited to check if a sequence ρ includes the behavior of interest. Indeed, it suffices to check if it is possible to extract from the event sequence $\rho \in \mathcal{T}(\Sigma)$ an ordered set of events that is a word of the language $\mathcal{R}$ generated by the pattern. In this case we say that the sequence ρ is $\mathcal{R}$-matching.

Definition 2.4. ($\mathcal{R}$-matching). Let $\mathcal{R}$ be a timed pattern, a sequence $\rho \in \mathcal{T}(\Sigma)$ is $\mathcal{R}$-matching if there exists a sequence $\omega \in \mathcal{R}$ such that $\rho \Supset \omega$ (i.e ρ matches ω).

2.3 Timed pattern diagnosis

Finally, we propose here to extend the original fault diagnosis problem to consider more complex behaviors than simple faults: behaviors described by timed patterns. For a given behavior pattern $\mathcal{R}$, the diagnosis problem is to define a $\mathcal{R}$-diagnoser function that takes as input a sequence of observations and returns one of the three symbols:
$\{\mathcal{R}-certain, \mathcal{R}-absent, \mathcal{R}-ambiguous\}$

Definition 2.5. ($\mathcal{R}$-diagnoser). Given $\mathcal{L}$ the timed language of the system over the event set $\Sigma = \Sigma_u \cup \Sigma_o$, an $\mathcal{R}$-diagnoser is a function

$$\Delta_{\mathcal{R}} : \mathcal{T}(\Sigma_o) \to \{\mathcal{R}-certain, \mathcal{R}-absent, \mathcal{R}-ambiguous\}$$

such that:

- $\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-certain$ if for any sequence $\rho \in \mathcal{L}$ such that $P^t_{\Sigma_o}(\rho) = \sigma$, ρ is $\mathcal{R}$-matching;

- $\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-absent$ if for any sequence $\rho \in \mathcal{L}$ such that $P^t_{\Sigma_o}(\rho) = \sigma$, ρ is not $\mathcal{R}$-matching;
- $\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-ambiguous$ otherwise.

Considering a set of patterns $\mathcal{R}_1, \ldots, \mathcal{R}_n$ the diagnoser function is defined by:

$$\Delta : \mathcal{T}(\Sigma_o) \to \prod_{i=1}^{n}\{\mathcal{R}_i - certain, \mathcal{R}_i - absent, \mathcal{R}_i - ambiguous\}$$

such that $\Delta(\sigma) = (\Delta_{\mathcal{R}_1}(\sigma), \ldots, \Delta_{\mathcal{R}_n}(\sigma))$.

3. DIAGNOSIS BY MODEL CHECKING

The proposed method defines the problem of timed pattern diagnosis as a problem of reachability and uses model checking technics to solve it. This resolution relies on the TINA toolbox (TIme Petri Net Analyzer) (Berthomieu et al. (2004)). Given a system and a pattern, the method proceeds as follows.

(1) The timed languages of the system and of the pattern are modeled by L-type Labeled Time Petri nets respectively denoted Θ and Ω.
(2) From the two previous nets, a new net is built (Θ_Ω) that captures the matching relation between the system and the pattern.
(3) The input observation sequence is transformed into an L-type Labeled Time Petri net (denoted Obs).
(4) The result (Θ_Ω) is then synchronised with the Petri net model of the observation sequence (Obs) to obtain the net ($\Theta_\Omega || Obs$).
(5) The model checker of TINA is used to verify the matching property between the system and the pattern according to the input observation sequence on the net ($\Theta_\Omega || Obs$).
(6) The diagnosis result is thus given according to the answer of the model checker.

All the steps are detailed hereafter.

3.1 Timed languages modeling

The approach relies on the following class of Time Petri Nets (see Berthomieu et al. (2006) and Peterson (1977)).

Definition 3.1. (LLTPNPr). An *L-type Labeled Time Petri Net with Priorities* (LLTPNPr for short) is a tuple $\mathcal{N} = \langle P, T, A, \succ, \ell, L, \Sigma, I_s, Q, M_0\rangle$ with:

- P: a finite set of nodes called places;
- T: a finite set of nodes called transitions and $P \cap T = \varnothing$;
- $A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs between nodes.
- $\succ \subset T \times T$: the priority relation, not reflexive, not symetrical, and not transitive; $(t_1 \succ t_2)$ means that if t_1 and t_2 are both enabled then only t_1 is firable.
- $\ell : P \cup T \to L \cup \Sigma \cup \{\lambda\}$: the deterministic labelling function where L (resp. Σ) is the set of place labels (resp.transition labels); For $r = \delta_1 t_1 \delta_2 t_2 \cdots \in (\mathbb{N} \times T)^\star$, $\ell(r) = \delta_1 \ell(t_1) \delta_2 \ell(t_2) \ldots$;
- $I_s : T \to I^+$: the static time interval function, where I^+ is the non empty set of time intervals with non negative rational bounds.
- Q: the set of final markings and M_0 the initial marking.

A marking M is a function from P to $\mathbb{N}$ which maps each place with a number of tokens. For an LLTPNPr $\mathcal{N}$, the functions $\mathrm{pre}(t) = \{p \in P : (p,t) \in A\}$ and $\mathrm{post}(t) = \{p \in P : (t,p) \in A\}$ are defined. A state of $\mathcal{N}$ is a couple $\langle M, I\rangle$, with: M a marking and I the firing interval of the transition .

Priorities ($\succ$) aim at managing transition conflicts and modify the classical transition firing rules. Transition t is firable from state $\langle M, I\rangle$ after a duration δ, denoted $\langle M, I\rangle \to [\delta t]$, if:

(1) $M \geqslant \mathrm{pre}(t)$;
(2) $\delta \in I(t)$;
(3) $\forall t' \neq t : M \geqslant pre(t') \implies \delta \leq \sup(I(t'))$;
(4) $\forall t' \in T \succ t : M \ngeqslant \mathrm{pre}(t') \vee \delta \notin I(t')$.

Intuitively, t is firable after δ time units from state $\langle M, I\rangle$, if (1) t is enabled, (2) δ belongs to the time interval associated with t and is not greater than the upper bound of any other enabled transition t', (3) at time δ, t has the highest priority among the transitions that could be fired without the defined priorities. $\mathrm{R}(\mathcal{N}, M_0, I_0)$ is the set of reachable states from $\langle M_0, I_0\rangle$, with M_0 the initial marking and I_0 the restriction of the static time interval function to the enabled transitions at M_0. As a L-type net (see Peterson (1977)), an LLTPNPr $\mathcal{N}$ generates the timed language $\mathcal{L}(\mathcal{N})$ that is the set of timed sequences issued from the transition sequences generated by the net and leading to a final marking $M \in Q \cap \mathrm{R}(\mathcal{N}, M_0, I_0)$.

Definition 3.2. (Timed system model). A timed system of timed language $\mathcal{L}$ is an LLTPNPr $\Theta = \langle P_\Theta, T_\Theta, A_\Theta, \succ_\Theta, \ell_\Theta, L_\Theta, \Sigma_\Theta, I_{s\Theta}, Q_\Theta, M_{0\Theta}\rangle$ such that Q_Θ is the set of reachable markings and $\mathcal{L} = \mathcal{L}(\Theta)$.

Figure 5 presents an LLTPNPr model of a system used in the application section (see section 4).

Definition 3.3. (Pattern model). A pattern of timed language $\mathcal{R}$ is an LLTPNPr $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \succ_\Omega, \ell_\Omega, L_\Omega, \Sigma_\Omega, I_{s\Omega}, Q_\Omega, M_{0\Omega}\rangle$ such that any final marking of the set Q_Ω is a dead marking and $\mathcal{R} = \mathcal{L}(\Omega)$ (see definition 2.3).

Definition 3.3 extends the pattern model of Gougam et al. (2017) with time. Several class of patterns and systems can be considered (see Figure 2 for some examples).

3.2 Timed pattern diagnosis of timed workflows

The general problem of timed pattern diagnosis in timed system as defined above may lead to decidability issues (Popova-Zeugmann (2013)). To remain computable, we now restrict the problem to a subclass of timed systems that are *timed workflows*.

Definition 3.4. (Timed workflow). A timed workflow Θ is a timed system where any event $e \in \Sigma_\Theta$ occurs at most once in a run of the workflow.

Such a timed system can be represented by an acyclic and safe Time Petri net. Also, to remain computable, a pattern Ω is considered safe and conflict-free and must describe a behavior that is not reduced to the elapsed time (i.e the sequence 0λ is not part of $\mathcal{L}(\Omega)$ which means $M_{0\Omega} \notin Q_\Omega$). Finally, to keep the causality relation described by the underlying untimed Petri net, any transition not enabled by the initial marking has a static time interval with a strictly positive bound.

3.3 A model to capture the matching relation

The diagnosis function defined in Section 2.3 is based on the matching relation between the evolutions of the system that are consistent with the input observation sequence and the pattern. To capture this relation, Θ and Ω models are composed to obtain a new Petri net model denoted Θ_Ω. This

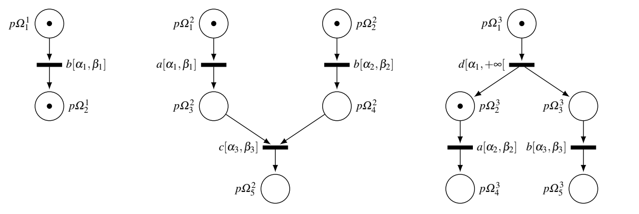

Figure 2. Three patterns Ω_1, Ω_2 and Ω_3 with $Q_{\Omega_1} = \{\{p_{\Omega_2^1}\}\}$, $Q_{\Omega_2} = \{\{p_{\Omega_5^2}\}\}$ and $Q_{\Omega_3} = \{\{p_{\Omega_4^3}, p_{\Omega_5^3}\}\}$.

model represents the synchronized evolutions of the system and the pattern but also the evolutions of the system alone as a pattern evolution corresponds potentially (in case of matching) to a sub-word of the system. The combination of both nets is obtained by performing the union of the system places and the pattern places. The transitions of the system that do not share any event with the pattern are kept with their initial time intervals. The other ones must be synchronized with the pattern's transitions. As in Time Petri nets compositionnality is kept only when synchronizations involve transitions with an interval $[0, +\infty[$ (Berthomieu et al. (2006)), we need to proceed to a time decomposition of the transitions to synchronize (see Figure 3). Transition t_1 (on the left) is labeled by a with a finite time interval $[\alpha, \beta]$. t_1 is replaced by a sub-net (on the right) composed by a silent transition (labeled by λ not indicated on the figure) of interval $[\alpha,\alpha]$ whose firing enables both a transition labeled by a (t_3) with the interval $[0, +\infty[$ and a silent transition (t_2) of interval $[\beta - \alpha + 1, \beta - \alpha + 1]$. At $\beta + 1$ (i.e β times unit + one time unit) as this transition has no priority, the silent transition (t_2) is fired leading to an unacceptable behavior represented by the marking of a specific place $unAcc$. This transformation applies to transitions of the system as well as transitions of the pattern. Therefore in the system the specific place is named sys_unAcc whereas in the pattern it is pat_unAcc.

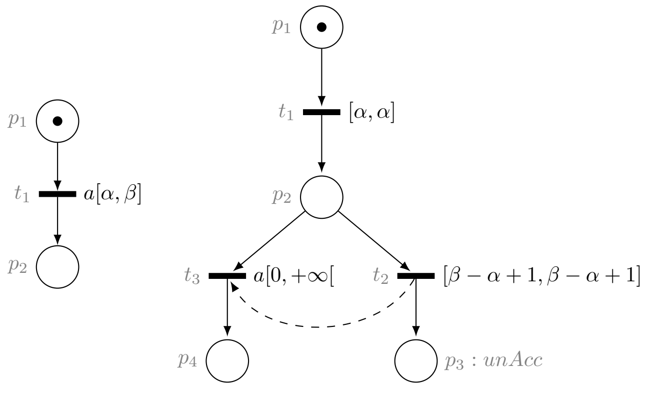

Figure 3. Transformation of a transition labeled by a with a finite interval $[\alpha, \beta]$: on the left the transition on the right the sub-net of replacement.

Given a system Θ and a pattern Ω, each reachable marking M in Θ_Ω is then the union of two markings, one is reachable in Θ, and the other one is reachable in Ω. $\forall \langle M, I\rangle \in \mathrm{R}(\Theta_\Omega, \langle M_{0\Theta_\Omega}, I_{0\Theta_\Omega}\rangle) : \exists \langle M_1, I_1\rangle \in \mathrm{R}(\Theta, \langle M_{0\Theta}, I_{0\Theta}\rangle)$, $\exists \langle M_2, I_2\rangle \in \mathrm{R}(\Omega, \langle M_{0\,\Omega}, I_{0\,\Omega}\rangle) : M = M_1 \cup M_2$. Moreover $\mathcal{L}(\Theta_\Omega) = \{\rho \in \mathcal{L}(\Theta) : \exists \omega \in \mathcal{L}(\Omega) : \rho \Supset \omega\}$.

Finally Figure 4 illustrates the complete combination in Θ_Ω of a system transition labeled by b and assigned with an interval $[1, 4]$ with a pattern transition labeled by b with an interval

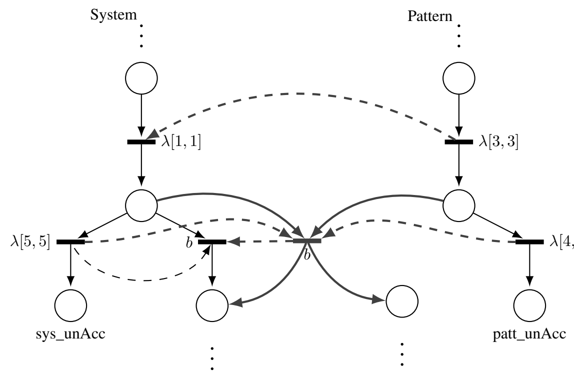

Figure 4. Part of Θ_Ω: combination of a Θ-system transition ($b[1, 4]$) with a Ω-pattern transition ($b[3, 6]$).

$[3, 6]$. Two transitions labeled by b appear: the one of the right results from the synchronization of the system and the pattern transitions and the left one represents the standalone evolution of the system. Priorities (dashed lines) ensure that the synchronized transition is triggered just in case of matching. For each possible run of the system, the net Θ_Ω represents whether this run matches the pattern or not. [1] Finally, to solve the diagnosis problem, this model must now be confronted with a sequence of observations.

3.4 Integration of the observation sequence

The observation sequence is first modeled by an L-Type Labeled Time Petri net denoted Obs. For instance for an observation sequence $1o_13o_2$, Obs has one transition t_1 labeled o_1 assigned with a static time interval $[1, 1]$. The output place of this transition enables a second transition t_2 labeled by o_2 with an interval $[3, 3]$. The output place of t_2 is denoted p_{Obs}. Afterwards, the combination system-pattern (i.e the net Θ_Ω) is synchronised (Hack (1975)) with the Petri model of the observation sequence (Obs) to obtain the net ($\Theta_\Omega||Obs$). To perform the synchronization the transitions of the observation sequence net (Obs) and the transitions of the combination (Θ_Ω) are transformed so that the synchronizations involve only transitions with $[0, +\infty[$ intervals as explained in Section 3.3. The verification of the matching property between the system and the pattern according to the input observation sequence on $\Theta_\Omega||Obs$ is then performed by the model checker. Then the diagnosis function ($\mathcal{R}$-diagnoser) generates its results according to the answers of the model checker as presented now.

3.5 Implementation of the $\mathcal{R}$-diagnoser function by Model checking

Back to the description of the method in Section 3, the overall computational complexity of the method lies in step 5 and we propose to use an *off-the-shelf* model checking tool to solve this part: the TINA toolkit (Berthomieu et al. (2004)). TINA generates *enriched labeled Kripke structures* from which it is possible to check properties written in SE-LTL (State/Event Linear Temporal Logic) which extends LTL in the way presented hereafter. A formula ψ is a SE-LTL formula if it is a universally quantified formula : $\psi ::= \forall \varphi$ such that:

[1] This property is true because the systems that are considered here are workflows (see Definition 3.4).

$\varphi ::= cst \mid r \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi\mathbf{U}\varphi$
$r ::= e \mid e \bigtriangleup e$
$e ::= p \mid a \mid c \mid e \bigtriangledown e$

with p a place symbol, a a transition symbol, $c \in \mathbb{N}$, $\bigtriangleup \in \{=, <, >, \leq, \geq\}$ and $\bigtriangledown \in \{+, -, *, /\}$. The operators $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually) and $\mathbf{U}$ (until) have their usual LTL semantics.

The $\mathcal{R}$-diagnoser is based on two questions written in SE-LTL concerning the Petri net model $\Theta_\Omega || Obs$:

(1) Question $\varphi_{CERTAIN}$: Are we sure that the pattern is recognized in each system's behavior that produces the observations ? If the result is false the following question is asked to the model checker:
(2) Question φ_{ABSENT}: Are we sure that the pattern is never recognized in each system's behavior that produces the observations ?

More formally, each question corresponds to a SE-LTL formula. Given M a marking of $Obs \parallel \Theta_\Omega$, $M_{|\Omega}$ denotes the restriction of M to Ω i.e. the marking of the pattern's places.

(1) $\varphi_{CERTAIN} \equiv \Box((sys\_unAcc = 0 \wedge p_{Obs} = 1) \Rightarrow M_{|\Omega} \in Q_\Omega)$, in others words, is it always true ($\Box$) that if the system generates *Obs* ($sys\_unAcc = 0 \wedge p_{Obs} = 1$) then ($\Rightarrow$) it matches the pattern ($M_{|\Omega} \in Q_\Omega$).
(2) $\varphi_{ABSENT} \equiv \Box((sys\_unAcc = 0 \wedge p_{Obs} = 1) \Rightarrow M_{|\Omega} \notin Q_\Omega)$, in others words, is it always true ($\Box$) that if the system generates *Obs* ($sys\_unAcc = 0 \wedge p_{Obs} = 1$) then ($\Rightarrow$) it does not match the pattern ($M_{|\Omega} \notin Q_\Omega$).

4. APPLICATION TO THE DIAGNOSIS OF A TIMED WORKFLOW OF A FOODSHOP.

The proposed method is illustrated on an example denoted as the *foodshop* example. This example is inspired from the Web services' systems that were studied in the European project WSDIAMOND Di Nitto et al. (2009). Figure 5 presents the workflow of a foodshop service. In this type of application, fault patterns usually represent misbehaviors not compatible with the service specification instead of physical faults on hardware. In this workflow, a foodshop first receives the order of a client (*sh_recvg*). Within the next 3 hours, the shop must start two activities (*sh_st[1,2]*) in parallel. The first activity dispatches the order into two categories: the storable products (not perishable) and the non-storable products (perishable ones). This dispatch activity ends with the event *sh_end_spl*. The second activity selects a warehouse. For storable products, the shop queries the warehouse to check the availability of the products (*sh_req_whcheck*). The warehouse then checks (*wh_check*) and provides a feedback (available *wh_send_av*, unavailable *wh_send_unav*). For non-storable products, the shop sends directly a request to a supplier (*sh_req_suppcheck*) that also provide feedbacks (*sp_send_av*, *sp_send_unav*). The shop requires that the feedbacks are given within the next 4 hours. After the reception of the feedbacks, the shop decides to implement or not the order based on the availabilities (*sh_decide*). If one product is missing the shop cancels the order (*sh_cancel*), if not the shop requests the preparation for delivery to the warehouse and the supplier (*sh_req_deliv*). When preparations are finished (*wh_end_prep_deliv* and *sp_end_prep_deliv*), it is finally delivered. In Figure 5 (on the left), observable events are in bold. They are events from the shop that manages the whole delivery process. To illustrate the method, we consider two patterns.

4.1 Occurrence of *sp_send_unav*

This pattern is the occurrence of *sp_send_unav*. This pattern models the fact that the supplier sends as feedback that the goods are not available. This simple pattern aims at illustrating that our method extends the classical problem where fault are modeled by single events. This pattern is composed of two places pp_0, pp_1 and one transition. Initial marking is $M_0(pp_0) = 1$ and $M_0(pp_1) = 0$ and the only accepting marking $M \in Q_\Omega$ is such that $M(pp_0) = 0$ and $M(pp_1) = 1$. Consider one run of the timed workflow and suppose that it produces the observable sequence σ_1 whose conversion to Petri net is shown in Figure 5 (on the right). To solve the problem, we first build the product between the system and the pattern that is then synchronized with the Petri net of σ_1. On the resulting Petri net, we first query to check whether any run of the system that can produce σ_1 matches the pattern with the following query $\varphi_{CERTAIN}$:

```
[] ((sys_unAcc = 0 /\ p9=1) => (pp1=1))
```

TINA's response is negative: there exists at least one run of the system that produces σ_1 and for which *sp_send_unav* has not happened. Then we check whether these runs do not match (i.e. contain) *sp_send_unav* with the formula φ_{ABSENT}:

```
[] ((sys_unAcc = 0 /\ p9=1) => -(pp1=1))
```

This is a positive answer. So, the diagnosis of σ_1 is that *sp_send_unav* did not happen. Let us consider now the observable sequence σ_2 composed of the first five events of σ_1 followed by **8***sh_decide* **2***sh_cancel*. In this case $\varphi_{CERTAIN}$ is true: any run generating σ_2 contains *sp_send_unav*. Querying TINA with φ_{ABSENT} is useless: the diagnosis is certain.

4.2 Occurrence of a sequence pattern

Consider *wh_send_av*$[0, +\infty[\rightarrow$ *wh_end_prep_deliv*$[70, 200]$. This second pattern illustrates a misbehaviour: if the warehouse sends as a feedback that the goods are available then the preparation ends within 70 and 200 time units which is considered as an unacceptable duration (quality of services is impacted). Let us consider the following observed sequence: $\sigma_3 =$ **50***sh_recvg* **1***sh_st* **1***sh_end_spl* **1***sh_req_whcheck* **1***sh_req_suppcheck* **3***sh_decide* **2***sh_req_deliv* **50***deliv*. TINA concludes that the pattern has not happened. This is due to the fact that the delivery date is time 50 after the order request. Now if we modify σ_3 by increasing the date 50 to 80, the diagnosis becomes certain. Only the observation of time is discriminating here.

5. CONCLUSIONS ET PERSPECTIVES

This work extends the original problem of the diagnosis of simple faults to the diagnosis of timed patterns in timed systems. The diagnoser function is based on the notion of matching between the system evolutions that are consistent with an input sequence of observations and the pattern. A resolution of the problem based model checking techniques is proposed to solve the problem on timed workflows. This work must be completed by a diagnosability analysis inspired from Gougam et al. (2017). We are also planning to investigate other subclasses of Time Petri nets where the problem is decidable.

REFERENCES

Basile, F., Chiacchio, P., and De Tommasi, G. (2007). Improving on-line fault diagnosis for discrete event systems using

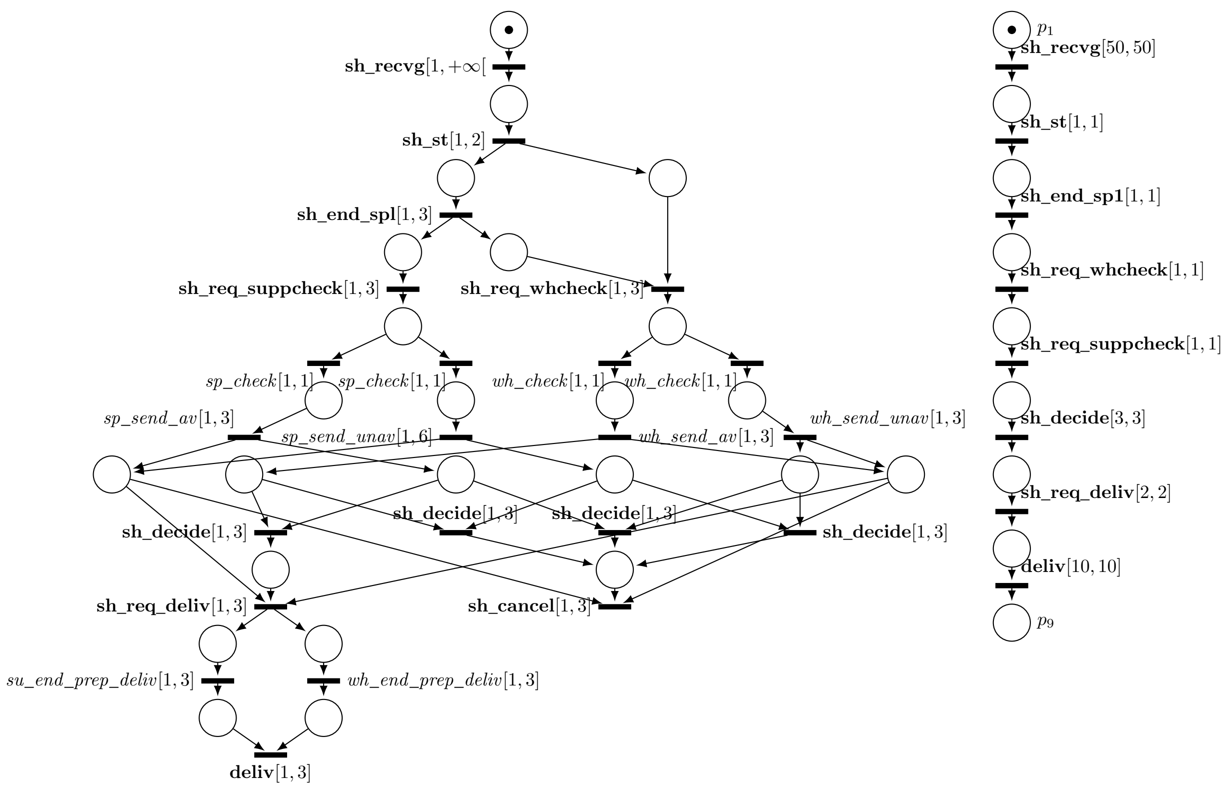

Figure 5. *Foodshop* timed workflow and the sequence of observations $\sigma_1 = \mathbf{50}$*sh_recvg* $\mathbf{1}$*sh_st* $\mathbf{1}$*sh_end_spl* $\mathbf{1}$*sh_req_whcheck* $\mathbf{1}$*sh_req_suppcheck* $\mathbf{3}$*sh_decide* $\mathbf{2}$*sh_req_deliv* $\mathbf{10}$*deliv*.

time. In *3rd IEEE Conference on Automation Science and Engineering (CASE'07)*, 26–32. Scottsdale, Arizona.

Basile, F., Cabasino, M.P., and Seatzu, C. (2015). State estimation and fault diagnosis of labeled time petri net systems with unobservable transitions. *IEEE Trans. Automat. Contr.*, 60(4), 997–1009.

Berthomieu, B., Peres, F., and Vernadat, F. (2006). Bridging the gap between timed automata and bounded time Petri nets. In *4th International Conference Formal Modeling and Analysis of Timed Systems*, 82–97. Paris, France.

Berthomieu, B., Ribet, P.O., and Vernadat, F. (2004). The tool tina – construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14), 2741–2756.

Boel, R.K. and Jiroveanu, G. (2013). *The On-Line Diagnosis of Time Petri Nets*, 343–364. Springer London, London.

Bonhomme, P. (2015). Fault diagnosis of p-time labeled petri net systems. In *Proceedings of the 9th Workshop on Verification and Evaluation of Computer and Communication Systems, VECoS 2015, Bucharest, Romania, September 10-11, 2015*, 11–22.

Bozzano, M., Cimatti, A., Gario, M., and Tonetta, S. (2015). Formal design of asynchronous fault detection and identification components using temporal epistemic logic. *Logical Methods in Computer Science*, Volume 11, Issue 4.

Chatain, T. and Jard, C. (2005). Time supervision of concurrent systems using symbolic unfoldings of time petri nets. In *Third International Conference on Formal Modeling and Analysis of Timed Systems, Third International Conference, Uppsala, Sweden*, 196–210.

Di Nitto, E., Sassen, A.M., and Traverso, P. (2009). *WS-DIAMOND Web Services DIAgnosability, MONitoring, and Diagnosis*, 213–240. MIT PRESS.

Gougam, H.E., Pencolé, Y., and Subias, A. (2017). Diagnosability analysis of patterns on bounded labeled prioritized Petri nets. *Discrete Event Dynamic Systems*, 27(1), 143–180.

Hack, M. (1975). Petri net languages. Technical Report 124, M.I.T. Project MAC, Computation Structures Group, Massachusetts Institute of Technology.

Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *8th International Workshop on Discrete Event Systems*, 262–268. Ann Arbor, MI, United States.

Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *Transactions on Automatic Control*, 49(6), 934–945.

Jiroveanu, G. and Boel, R.K. (2006). A distributed approach for fault detection and diagnosis based on time petri nets. *Mathematics and Computers in Simulation*, 70(5-6), 287–313.

Liu, B. (2014). *An efficient approach for diagnosability and diagnosis of DES based on labeled Petri nets - untimed and timed contexts*. Ph.D. thesis, Univ. Lille Nord.

Peterson, J.L. (1977). Petri nets. *ACM Computing Surveys*, 9(3), 223–252.

Popova-Zeugmann, L. (2013). *Time and Petri Nets*. Springer.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *Transactions on Automatic Control*, 40(9), 1555–1575.

Tripakis, S. (2002). Fault diagnosis for timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, Oldenburg, Germany, September 9-12*, 205–224.

Wang, X., Mahulea, C., and Silva, M. (2015). Diagnosis of time petri nets using fault diagnosis graph. *IEEE Transactions on Automatic Control*, 60(9), 2321–2335.