On quantitative Analysis of Time Open Workflow Nets and Parametric Extension

Zohra Sbaï
Université de Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis
BP. 37 Le Belvédère
1002 Tunis, Tunisia
zohra.sbai@enit.rnu.tn

Kamel Barkaoui
Laboratoire CEDRIC
Conservatoire National des Arts et Métiers
292 rue Saint Martin
Paris Cedex 03, France
kamel.barkaoui@cnam.fr

Collaborative systems design is today a complex process especially where constraints such as tasks delays or resources handling have to be considered. In addition to a well constructed model of a workflow system, a prior and rigorous verification phase is important to ensure a correct execution of the system. In this direction, we are interested in this paper by the modeling and the analysis of interacting processes in particular those constrained by timing delays. First, we present the Time open Workflow nets (ToWF-nets) which stand for a sub-class of Petri nets dedicated to model any business process which interact with other partners via interface places in order to meet a common goal. Then, after recalling the semantics of ToWF-nets, we investigate in presenting our recent results in quantitative verification of properties related to the correct communication of various ToWF-nets. We show how to make a TCTL based model checking of the studied properties. Finally, we extend our analysis approach of ToWF-nets to cover concurrent processes leading thus to propose a parametric verification of ToWF-nets.

Time Open Workflow Nets, Collaborative Systems, Quantitative Analysis, Timed Computational Tree Logic, Parametric Verification

1. INTRODUCTION

Nowadays, collaboration in organizations is more and more sought since it allows to end users to benefit from more enhanced and complex services. Although it is possible to allow complex services with a single organization, it is almost impossible to provide, in this case, simple services nor to reuse them. So, it is important to provide simple services and to allow their composition, if necessary, for more complex tasks. For instance, manufacturing systems can be seen as a network of servers and queues. This system process can be seen as a composition of multiple services interacting together to fulfill a unique goal. As an example, we mention a factory which make shirts. The first stage in shirt manufacturing is the cutting of the material to different shapes. Then, the next stage is to sew the pieces together. Finally buttons and a quick iron are added. This composition is more and more studied when treating inter organizational processes. We mention for example a manufacturing enterprize which has to communicate with clients, suppliers, etc. in order to fulfill the manufacturing task. Thus the notion of services composition is of great need and that's why it is very studied in academic as well as industrial environment.

In collaborative systems in general, different partners, having their own processes, interact together in order to achieve a common goal. We focus in this paper on the modeling and the analysis of such systems involving multiple communicating processes. In the modeling phase, we propose to use Time Open Workflow Nets (ToWF-nets), a sub class of Time Petri Nets (TPN) to model workflow processes with timing delays and with ability to communicate with partners. A ToWF-net consists of an open workflow net (oWF-net) in which we associate with each transition a minimum and a maximum amount of time needed to its execution. An oWF-net is a workflow net augmented with special places called interface places and used to interact with other processes.

For the purpose of analysis, we propose to enhance a formal verification due to solid theoretical basis of formal methods. More precisely, we adopt an analysis method based on the well known model checking techniques. In fact, Model checking is an automated verification technique for proving that a model satisfies a set of properties specified in

temporal logic. Given a concurrent system Σ and a temporal logic formula φ, the model checking problem is to decide whether Σ satisfies φ. Hence, we have to formulate in temporal logic the properties to be verified. This kind of verification is situated at the design phase, allowing thus to find bugs as early as possible and therefore to reduce the cost of failures. This, especially, permits us to check as early as possible if two or more processes are compatible before their composition. We express the proposed properties in Timed Computation Tree Logic (TCTL).

The rest of the paper is organized as follows. Section 2 is dedicated to the presentation of ToWF-nets for modeling interacting processes which are constrained by timing delays. In this section, we present the semantics of the model in terms of states and the states evolution. In section 3, we highlight quantitative analysis results of ToWF-nets. We recall first the principle of TPN-TCTL temporal logic, then we characterize some properties of interest and express them in this temporal logic and finally, we detail how to model check these properties via several examples. We focus in section 4 on a parametric verification of ToWF-nets composition and on some experiments in Romeo model checker. Finally, section 5 presents related work and section 6 concludes the paper with a discussion and a proposition of future work.

2. TIME OPEN WORKFLOW NETS

This section is dedicated to present Time open Workflow nets (ToWF-nets) : a Petri nets sub class allowing to model workflow processes communicating with partners via interface places. After defining ToWF-nets model and semantics, we expose some results on reachability analysis.

2.1. ToWF-nets to model communicating processes

To model a composition of interacting processes, we refer to Petri nets due to their expressive power as well as their theoretical basis. The well known Petri net class used in this modeling is named open workflow nets (oWF-nets) (Karsten and Schmidt (2005); Sbaï and Barkaoui (2013); Martens (2005); Massuthe et al. (2005)). Each involved process is modeled by an oWF-net possessing interface places which serve for the communication with other processes. In this way, the interaction and conversation between the different processes are guaranteed. The interaction considered here is ensured through operational and control behaviors. The operational behavior is specific to each process according to its business logic and the control behavior is ensured by the general behavior of any partner in the composite process.

As mentioned above, oWF-nets nets are mainly the extension of workflow nets (WF-nets) to model workflow processes which interact with other processes via interface places. Simple WF-nets (Sbaï and Barkaoui (2013, 2012)) is a result of Petri nets' application to workflow management. The choice of Petri nets is explained by the fact that they form a powerful formalism expressing the control flow in business processes (van der Aalst (1996); Esparza et al. (1989); Ellis et al. (1995); De Michelis et al. (1994)).

Commonly, a Petri net is a 4-tuple $N = (P, T, F, W)$ where *P* and *T* are two finite non-empty sets of places and transitions respectively, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is the weight function of *N* satisfying $W(x, y) = 0 \Leftrightarrow (x, y) \notin F$. If $W(u) = 1 \; \forall u \in F$ then *N* is called an ordinary net and it is denoted by $N = (P, T, F)$. For every node $x \in P \cup T$, the set of input nodes of x is defined by ${}^\bullet x = \{y | (y, x) \in F\}$ and the output nodes of x form a set denoted by $x^\bullet = \{y | (x, y) \in F\}$. We refer the reader to (Barkaoui et al. (2007)), for more Petri nets notations used in this paper.

A Petri net which models a workflow process is called a WF-net (van der Aalst (1997)). An ordinary Petri net $N = (P, T, F)$ is a WF-net iff N has one source place i named initial place (containing initially one token) and a sink place f named final place. In addition to this characteristic, every node in a WF-net $n \in P \cup T$ exists in a path from i to f.

For processes composition, we propose to model each process by a WF-net describing the tasks to be performed as well as their routing. Then, the conversation between the involved processes is ensured by the communication places used for the conversation and/or the messages sending. Thus, we are using open WF-nets (oWF-nets) which extend the classical WF-nets by integrating interface places to ensure asynchronous communication with the different partners. Hence, a composition is modeled by a set of oWF-nets interacting via interface places. Note that these places are used to connect only transitions from different processes.

Now when we focus on incorporating time constraints, we find different proposals which are based on extensions of Petri nets. In general, the time constraints are modeled either by durations or delays. When they are specified by durations (constants), the extension associated is said to be *Timed Petri nets*. These constant durations are either attached to places (the subclass is known as P-Timed Petri nets) or to transitions (T-Timed Petri nets). In case of delays adoption, the constraints are specified by means of intervals specifying the minimum and the

maximum amounts of time representing the transitions firing delays, the associated extension is called Time Petri nets. These intervals are attached to places (P-Time Petri nets), transitions (T-Time Petri nets) or arcs (A-Time Petri nets) leading thus to different extensions with variant semantics.

In the case of workflow processes, several studies (van der Aalst (1993); Atluri and Huang (1996); Ling and Schmidt (2000); MarteGou et al. (2001)) have shown the importance of temporal reasoning in the specification of workflow systems. In (Ling and Schmidt (2000)), the authors extend the simple WF-net presented by van der Aalst (van der Aalst (1997)) by associating with each transition a time amount representing the task duration. A temporal extension of the WF-net, called Timed WF-net is proposed. The semantic adopted here is that each enabled transition must fire immediately, otherwise the transition will be disabled. In case of firing, its duration should be respected, i.e. this transition can not be delayed. Although this approach is simple, it is strict in the sense that it requires fixed durations. To deal with this limitation, Time Workflow nets (TWF-nets) are proposed allowing to associate time delays with the activities by incorporating to each transition a time interval specifying its firing delay (Boucheneb and Barkaoui (2012); Camerzan (2007); Ling and Schmidt (2000)).

Since this time consideration is flexible and given that we are interested by modeling the composition of workflow processes with time constraints, we propose the Time open Workflow Net model (ToWF-net) (Sbaï et al. (2014)). This model associates to each transition a static time interval which refers to the execution time or delay of the corresponding activity. The formal definition of a ToWF-net model is the following:

Definition 1 ***(ToWF-net)***

A Time open Workflow net N *is a tuple* (P, T, F, FI, I, O) *with:*

- P *is a set of places,*
- T *is a set of transitions,*
- I *is a set of places which represent input interfaces that are responsible of messages receiving from other processes:* ${}^\bullet I = \emptyset$*.*
- O *is a set of places which represent output interfaces responsible of messages sending to other partners:* $O^\bullet = \emptyset$*.*
- I*,* O *and* P *are disjoint.* I *and* O *connect transitions of different processes.*
- $F \subseteq ((P \cup I) \times T) \cup (T \times (P \cup O))$ *is the flow relation,*
- $FI : T \to \mathbb{Q}^+ \times \mathbb{Q}^+ \cup \{\infty\}$ *is the function which associates to each transition* $t \in T$ *a static firing interval, i.e.* $FI(t) = [minFI(t), maxFI(t)]$ *where* $minFI(t)$ *and* $maxFI(t)$ *are rational numbers representing the minimal and the maximal firing time respectively,*

The marking of N is a vector of $\mathbb{N}^P$ such that for each place $p \in P$, $M(p)$ is the number of tokens in p. The initial marking of N is M_i knowing that M_p is used to denote a marking for which $M(p) = 1$ and $M(q) = 0\ \forall q \in (P \cup I \cup O) \setminus \{p\}$.

A transition t is enabled in a marking M if the required tokens are present in the input places of t. We denote by $En(M)$ the set of all the transitions enabled in the marking M. A transition t is said disabled by the firing of t' in M if it is enabled in M but it isn't in $M - {}^\bullet t'$. When focusing of newly enabled transitions after firing a transition t from M and leading to M', we denote by $NEn(M, t)$ the set of transitions enabled after this firing.

$$NEn(M, t) = \{t' \in En(M') | t' = t \vee \neg M \geq^\bullet t +^\bullet t'\}.$$

When a transition t becomes enabled, its firing interval is set to its static firing interval $FI(t)$. The upper and lower bounds of $FI(t)$ decrease synchronously with time, until t is fired or disabled by another firing. t can fire, if the lower bound of its firing interval reaches 0, but when upper bound of its firing interval reaches 0, t must be fired without any additional delay (strong semantic). The firing takes no time but may lead to another marking.

Let us define first the state of a ToWF-net and then the transition relation.

Definition 2 *A state in a ToWF-net is the state of the process that is modeled with ToWF-net after the occurrence of an event. Formally, we characterize a state in a ToWF-net by a pair* (M, Int) *where:*

- M *is a marking,*
- Int *is a firing interval function,* $Int : En(M) \to \mathbb{Q}^+ \times \mathbb{Q}^+ \cup \{\infty\}$*. We denote* $Int(t) = [minInt(t), maxInt(t)]$*.*

The initial state of a ToWF-net is (M_0, Int_0) where $M_0 = M_i$ (since in a ToWF-net, only i contains initially one token) and $Int_0(t) = FI(t)\ \forall t \in En(M_0)$.

Starting from the initial state (M_0, Int_0), the net evolves following the occurrence of events. An event corresponds to either a transition firing or a time progression. Hence, we define the transition relation

between states $s_1 = (M_1, Int_1)$ and $s_2 = (M_2, Int_2)$ by $\xrightarrow{d}$ in case of time progression and by $\xrightarrow{t}$ in case of a firing. We explicit in the following the conditions of state evolution and how to compute the resulting state after an event occurrence.

1. $s_1 \xrightarrow{t} s_2$ if and only if s_2 is immediately reachable from s_1 by firing the transition t, i.e.
   $t \in En(M_1)$ and $minInt_1(t) = 0$,
   $M_2 = M_1 -^\bullet t + t^\bullet$, and
   $\forall t' \in En(M_2), \quad Int_2(t') = \begin{cases} FI(t') \; if t' \in NEn(M_1, t) \\ Int_1(t') \quad otherwise \end{cases}$
2. $s_1 \xrightarrow{d} s_2 \; \forall d \in \mathbb{R}$ if and only if state s_2 is reachable from s_1 by a time progression with d time units, i.e.
   $minInt_1(t) + d \leq maxFI(t)$,
   $M_2 = M_1$, and
   $\forall t \in En(M_1)$, $Int_2(t) = [Max(0, minInt_1(t) - d), maxInt_1(t) - d]$

We can now define the semantics of a ToWF-net N by a transition system $(S, s_0, \rightarrow)$ with S the set of all the reachable states from the initial state s_0 by $\rightarrow$ the transition relation defined above.

We present in the following subsection some results of reachability analysis of ToWF-nets composition.

2.2. Reachability analysis of ToWF-nets

After the formal definition of ToWF-nets, we focus now on their reachability analysis. This analysis is based on the efficient construction of the state space.

By analogy with the marking graph defined in the context of an ordinary Petri net, we define a state space by a graph containing all accessible states of a ToWF-net from the initial state. Therefore, to calculate the state space of a ToWF-net, we must be able to calculate the reachable states by activating the enabled transitions.

Definition 3 *The state space of a ToWF-net has the following structure: $SS = (S, \rightarrow, s_0)$; where S is the set of nodes in form (M, Int) representing the reachable states from the initial one $s_0 = (M_i, Int_0)$; $\rightarrow$ represents the transition relation which defines the evolution from one state to another.*

$S = \{s | s_0 \xrightarrow{*} s\}$ is the set of reachable states of the model, and $\xrightarrow{*}$ is the reflexive and transitive closure of $\rightarrow$.

The reachability analysis (Boucheneb and Barkaoui (2012)) in timed models (such as time extensions of Petri nets as well as timed automata) is based in general on abstraction, which preserves reachability properties. Such an abstraction for timed models, consists in considering only one node for all states reachable from the same firing sequence while abstracting from their firing times. The grouped states, known as state classes, are then considered modulo some equivalence relation preserving properties of interest.

In return, the state class method is intended to provide a finite representation of the infinite state space of any bounded time Petri net.

Technical classes produce for a large class of time nets a finite representation of their behavior states, which allows a reachability analysis similar to that permitted for Petri nets by the technique of marking graph.

The state classes can be represented by a marking and a firing domain. Formally, a state class is a couple (M, D) where M is a marking and D is characterized by a set of atomic constraints over the firing delays of enabled transitions: $minFI(t) \leq t \leq maxFI(t) \;\; \forall t \in En(M)$.

Note that the initial class coincides with the initial state of the network. This initial class is (M_0, D_0) where $M_0 = M_i$ and D_0 corresponds to the firing domains of transitions enabled in M_0.

All states within the same node share the same untimed information and the union of their time domains is represented by a set of atomic constraints handled efficiently by means of a Difference Bound Matrix (DBM) (Ridi et al. (2012)). A DBM form a system of linear inequalities which constrain single variables $(v_1 ... v_n)$ and their differences within limits identified by coefficients b_{ij}. This is formally expressed as:

$$\begin{cases} v_i - v_j \leq b_{ij} \;\; i, j \in [0..n], \; b_{ij} \in \mathbb{Q} \\ v_0 = 0 \end{cases}$$

In terms of behavior, this state classes group preserves highly the states traces, and thus the safety properties.

The computation of the state class graph is necessary at this point to perform the various reachability analysis. Among the abstractions proposed in the literature (Berthomieu and Diaz (1991); Berthomieu and Vernadat (2003); Yoneda and Ryuba (1998)), we consider here the state class graph method (Berthomieu and Diaz (1991)) for its advantage, over the others, which is the finiteness property for all bounded time Petri nets (while using some approximations).

3. QUANTITATIVE ANALYSIS OF TOWF-NETS

The analysis of the state space is very significant to the extent that it can reveal important characteristics of the modeled system, about its structure and dynamic behavior. However, for a more accurate verification, we should not be limited to this type of checking rather than other specific properties. Indeed, we focus in this section on the formal verification of compatibility properties of ToWF-nets as an extension of our results of compatibility analysis while abstracting time information (Sbaï and Barkaoui (2014); Sbaï et al. (2014)). These properties focus on the correctness of the interactions between the different partners.

We propose to adopt model checking method to verify these properties since this method permits an exhaustive verification over all the possible executions. Given a concurrent system Σ and a temporal logic formula φ, the model checking problem is to decide whether Σ satisfies φ. Hence, we have to formulate in temporal logic the properties to be verified.

3.1. Model checking TPN-TCTL

Real systems often have behaviors that depend on time. The ability to manipulate and model the temporal dimension of the events that take place in the real world is fundamental in many applications. These applications may involve banking, medical, or multimedia applications. The variety of applications motivate many recent studies that aim to integrate all the features necessary to take into account the time during verification.

TCTL (Timed Computational Tree Logic) is a timed extension of the temporal logic CTL. TCTL added to CTL a quantitative information on the delays between actions. It is built from atomic propositions, logical connectors and temporal operators (U, F, G, X, etc.). The TCTL temporal logic can be used to check the properties of a time Petri net.

The syntax of TCTL formulas is inductively defined by:

φ ::= *false* | $\neg\varphi$ | $\varphi \wedge \varphi$ | *A*(φ U_I φ)| *E*(φ U_I φ)

where p denotes a proposition, φ denotes a formula and $I = [a, b]$ or $[a, \infty[$ with $a \in \mathbb{N}$ and $b \in \mathbb{N}$.

A and E are temporal quantifiers over the set of executions. $A\varphi$ announces that all the executions from the current state satisfy the property φ. $E\varphi$ states that from the current state, there exists an execution which satisfies φ. Finally $\varphi\ U_I\psi$ means that the property φ is true until ψ is true, and ψ will be true in the time interval I.

We can use other compositional temporal operators (Alur et al. (1993)): $EF_I\ \varphi$ = E(true $U_I\ \varphi$) (Possibility), $EG_I\ \varphi = \neg$ AF $_I\ \neg\varphi$ (All locations along an execution), $AF_I\ \varphi$ = A(true $U_I\ \varphi$) (Locations along all executions), $AG_I\ \varphi = \neg$ EF $_I\ \neg\varphi$ (All locations along all executions).

Semantically, TCTL formulas are interpreted on states and execution paths of a model $M = (S, V)$ where S is a transition system and V is a valuation function that associates with each state the set of atomic propositions it satisfies. (Konur et al. (2013))

To interpret a TCTL formula on an execution path, we introduce the notion of dense execution path. Let $s \in S$ be a state of S, $\pi(s)$ the set of all execution paths starting from s, and $\rho = s_0 \overset{d_0 t_0}{\rightarrow} s_1 \overset{d_1 t_1}{\rightarrow} s_2...$ an execution path of s. The dense path of the execution path ρ is the mapping $\hat{\rho} : R^+ \rightarrow S$ defined by: $\hat{\rho}(r) = s^i + \delta$ such that $r = \sum_{j=0}^{i-1} d_j + \delta$, $i \geq 0$ and $0 \leq \delta \leq d_i$.

The formal semantics of TCTL is given by the satisfaction relation defined as follows:

- $M, s \not\vDash false$,
- $M, s \vDash \phi$ iff $\phi \in V(s)$,
- $M, s \vDash \neg\varphi$ iff $M, s \not\vDash \varphi$,
- $M, s \vDash \varphi \wedge \psi$ iff $M, s \vDash \varphi$ and $M, s \vDash \psi$,
- $M, s \vDash \forall(\varphi \cup_I \psi)$ iff $\forall\rho \in \pi(s)\ \exists r \in I, M, \hat{\rho}(r) \vDash \psi$ and
  $\forall 0 \leq r' \leq r\ M, \hat{\rho}(r') \vDash \varphi$,
- $M, s \vDash \exists(\varphi \cup_I \psi)$ iff $\exists\rho \in \pi(s)\ \exists r \in I, M, \hat{\rho}(r) \vDash \psi$ and
  $\forall 0 \leq r' \leq r\ M, \hat{\rho}(r') \vDash \varphi$,

When interval I is omitted, its value is by default $[0, \infty[$.

The Time Petri net model is said to satisfy a TCTL formula φ iff $M,\ s_0 \vDash \varphi$.

The logic TCTL allows writing temporal properties with a quantification of the time. We chose this approach because it is decidable and PSPACE-complete for bounded Petri nets (Boucheneb et al. (2009)).

The authors of (Hadjidj and Boucheneb (2009)) have gone further by defining a sub-class of TCTL for time Petri nets in dense time, called TPN-TCTL. They proved the decidability of model-checking of TPN-TCTL on Petri nets and showed that its complexity is PSPACE.

Definition 4 *The temporal logic TPN-TCTL is defined inductively by:*

TPN-TCTL ::= false | φ | $\neg\varphi$ | $\varphi \vee \psi$ | $\varphi \wedge \psi$ | $\varphi \Rightarrow \psi$ | $E\varphi U_I \psi$ | $A\varphi U_I \psi$

| $EG_I\varphi$ | $AG_I\varphi$ | $AF_I\varphi$ | $EF_I\varphi$ | $AG(\phi_1 \Rightarrow AF_{[0,d]}\phi_2)$.

Where φ and $\psi \in$ TPN-TCTL,

$I = [a, b]$ or $[a, b[$ with $a \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\infty\}$.

ϕ_1 and ϕ_2 are propositions on markings.

$\forall G(\phi_1 \Rightarrow \forall F_{[0,d]}\phi_2)$ means that from the current state, any occurrence of marking ϕ_1 is followed by an occurrence of marking ϕ_2 less of d units of time later.

Romeo permits a practical implementation of the verification of properties described in TPN-TCTL. It is therefore possible to model check on the fly temporal quantitative properties. That's why we investigate in the following section the TCTL expression of the compatibility property and hence its verification in Romeo.

Before this, let us recall the notation used by Romeo to implement a TPN-TCTL property:

TPN-TCTL$_{Romeo}$ = $E(p)U[a,b](q)$ | $A(p)U[a,b](q)$ | $EF[a,b](p)$ | $AF[a,b](p)$ | $EG[a,b](p)$ | $AG[a,b](p)$ | $EF[a,b](p)$ | $(p) \rightarrow [0,b](q)$.

where p, q: GMEC; U: until; E: exists; A: forall; F: eventually; G: always; $\rightarrow$: response; a: integer; b integer or inf (to denote ∞).

$GMEC = a{*}M(i)\{+,-\}b{*}M(j)\{<,<=,>,>=,=\}k$ | $deadlock$ | $bounded(k)$ | p *and* q | p *or* q | $p \Rightarrow q$ | $not\ p$.

M: keyword (marking); $deadlock, bounded$: keywords; i, j:place indexes; a, b, k :integers ; $*, +, -, and, or, \Rightarrow, not$: usual operators ; p, q: GMEC

The syntax $(p) \rightarrow [0,b](q)$ denotes a leads to property meaning $AG((p)$ imply $AE[0,b](q))$. E.g. $(p) \rightarrow [0,b](q)$ holds iff whenever p holds eventually q will hold as well in $[0, b]$ time units.

3.2. TCTL characterization of ToWF-nets compatibility

In a composition of two or more processes, the correctness of the composite process refers in general to the compatibility of the different processes. From a behavioral point of view, the involved processes are compatible if they can interact properly. This means that composite process does not suffer from any deadlock problem. There is a distinction between syntactic compatibility which refer to the conformance of interfaces (number of interfaces, names, input, ouptut, etc.) and semantic compatibility which refer to the absence of deadlocks in the global system. In this paper, we focus only on defining and verifying the semantic compatibility.

Before this, let us characterize the composite system obtained by superposing a number of ToWF-nets which are supposed syntactically compatible. The composite net N of nbX ToWF-nets N_1... N_{nbX} consists of all ToWF-nets sharing interface places. In this composite net, every input interface of a TWF-net has to be an output interface place of another TWF-net of N. Trivially, N form a time Petri net with nbX output places and nbX input places. The initial marking of N is $M_0 = \sum_{s=1}^{nbX} i_s$.

In (Bordeaux et al. (2005); Foster et al. (2004); Martens (2003)), the authors characterized the compatibility of non timed services as the absence of deadlock in the composite service. They considered that two or more oWF-nets are compatible if they can (all) reach their final states. By analogy, we consider that two or more ToWF-nets are compatible if they reach their final states as well as the timing constraints are respected.

In order to relax this definition, we propose different categories of the compatibility property.

- Partial compatibility: A composite net N satisfies the partial compatibility if it is deadlock-free.
- Total Compatibility: A composite net N is totally compatible if it is deadlock-free and furthermore, the overall process terminates properly.
- Perfect compatibility: A composite net N is perfectly compatible if it is totally compatible and the deadline constraints are satisfied.
- Incompatibility: A composite net N is incompatible if it is neither partially nor totally compatible.

We focus in what follows on formulating the four types of compatibility properties. Let us consider the following notation:

- nbX: the number of processes;
- nbp: the number of places in a given process;
- nbi: the number of interface places in a composition;
- i_s: the input place of the process number s.
- f_s: the output place of the process number s.

Partial compatibility

Partial compatibility of nbX processes refers to the absence of deadlock in their composition. A net is deadlock-free if and only if there is at least a transition allowed in every marking except the final one M_{fin} which refer to the marking of all the final places f_s $(s = 1..nbX)$. This property is expressed as follows:

$$\forall M \in [M_0\rangle, M_{fin} \in [M\rangle$$

Logically, this deadlock-freeness property can be expressed as follows: "for all the executions made possible from the initial state, no deadlock is encountered until the final state is reached". The final state is characterized by the marking of the final places. Then, we can express the partial compatibility by the following TCTL formula:

$$AG_{[0,\infty[}((not\ MF) \ \Rightarrow \ not\ deadlock)$$

In this formula, $deadlock$ is a proposition supposed to return true if and only if there is no enabled transition in the current state. MF is a proposition on marking M_{fin} assuring that each final place is marked with at least one token.

$$MF = \overset{nbX}{\underset{s=1}{\wedge}} M(f_s) >= 1$$

Here we check only the marking of final places.

Total compatibility

The proper termination of a process refers to check if this latter complete its execution in any case, and at the time of termination, all the places of the involved ToWF-nets are empty except the final places which must be marked with exactly one token. Hence, we have to check if there exists a marking M for which all the places are empty except the output ones. Formally, this property can be expressed as follows:

$$\forall M \in [M_0\rangle :$$
$$M(f_s) \geq 1\ \forall s \in \{1, .., nbX\} \ \Rightarrow \ M = \textstyle\sum_{s=1}^{nbX} f_s$$

We can formulate the proper termination in TCTL as follows:

$$AF_{[0,\infty[}\ StrictMF$$

With $StrictMF$ a proposition on the marking with a token in every final place f_s but no tokens in the other places including the interface ones.

$$StrictMF = \overset{nbX}{\underset{s=1}{\wedge}} \ (\overset{nbip}{\underset{p=1}{\wedge}} (M(p) = 0)$$
$$\wedge \ (M(f_s) = 1)) \quad \wedge \quad (\overset{nbi}{\underset{i=1}{\wedge}} M(I_i) = 0)$$

where $nbip$ is used to denote the number of places other than the final place in a process.

Perfect compatibility

When we have a strict overall deadline that the system should respect, we can enforce the total compatibility by adding this constraint. We define, in this sense, the perfect compatibility which refers to both deadlock-freeness and proper termination while taking into account the deadline verification.

Let us suppose that the deadline constraint is considered, this can be checked by verifying that a process has to terminate (reach its final state) in Tm time units. Which lead to the expression of the proper termination within this delay as follows:

$$AF_{[0,Tm]}\ StrictMF$$

Hence, these two TCTL formulas can be used to express the perfect compatibility of ToWF-nets composition:

- $AG_{[0,Tm]}((not\ MF) \ \Rightarrow \ not\ deadlock)$
- $AF_{[0,Tm]}\ StrictMF$

Incompatibility

The incompatibility is a situation in which neither deadlock-freeness nor proper termination is verified. In other words, a set of ToWF-nets are incompatible if all the possible executions don't lead to final states of the different processes. We may distinguish here between strong incompatibility checked in the interval $[0, \infty[$ and weak incompatibility which refers to incompatibility in a limited interval $[a, b]$.

If we consider that the interval $[x, y]$ may correspond to $[0, \infty[$ or $[a, b]$, the expression of the incompatibility in TCTL leads to the following formula:

$$AG_{[x,y]}((not\ MF))$$

3.3. Model checking ToWF-nets composition

We present in this subsection some results related to the analysis of the compatibility and soundness properties for a ToWF-nets composition. This verification is ensured by Romeo (Gardey et al. (2005)) which is a software studio dedicated to time Petri nets analysis. Romeo is developed by the Real-Time Systems Team at IRCCyN. It performs analysis on Transition Time Petri Nets as well as on one of their extensions dedicated to scheduling. Romeo is chosen because it assures, among other features, a State Class Graph (SCG) computation and a graphical simulation of Transition Time Petri Nets. Moreover, Romeo is used here because it

implements an on the fly model checking algorithm of TPN-TCTL formulas.

We propose to study a simple composition of ToWF-nets (figure 1). We can easily verify that no deadlock will be encountered until final places are marked. Then the partial compatibility is satisfied (see figure 2). However, the firings of transitions T_4 and T_5 of the second ToWF-net lead to two tokens in its final place f_2; this violates the total compatibility property. In the figure 3, we show the result for this property and an execution trace. After that, we propose a correction in figure 4 for which the total compatibility is verified (see figure 5).

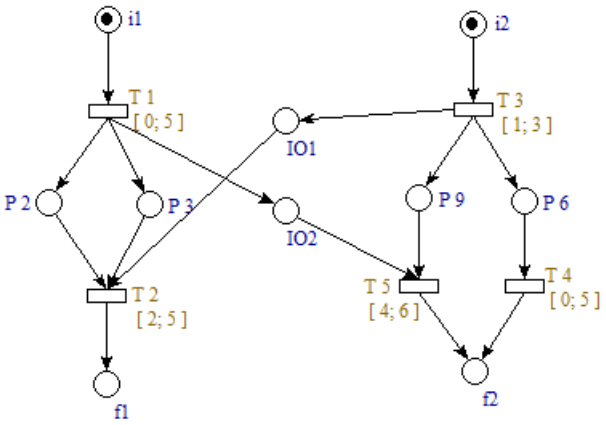

Figure 1: *Example 1 of ToWF-nets composition*

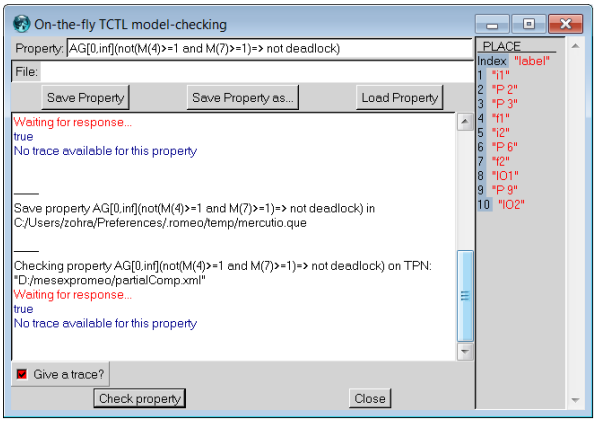

Figure 2: *Partial compatibility is verified*

4. PARAMETRIC ANALYSIS OF TOWF-NETS

In this section, we tackle with a parametric quantitative analysis of the composition of ToWF-nets. This suppose to consider k instances of each ToWF-nets ready to be executed.

For sake of covering the maximum number of real world processes, we tackle with time processes which share resource places. This places possess

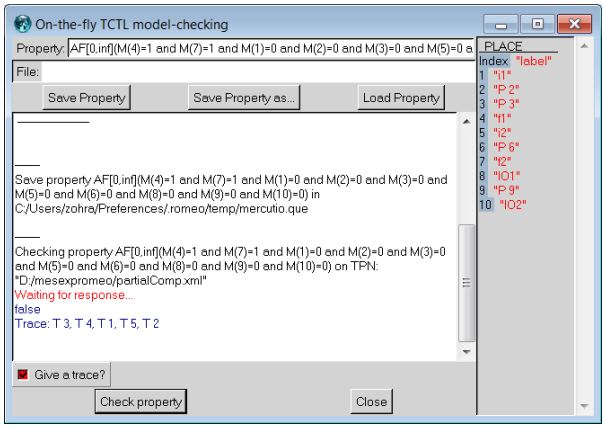

Figure 3: *Total compatibility is not verified*

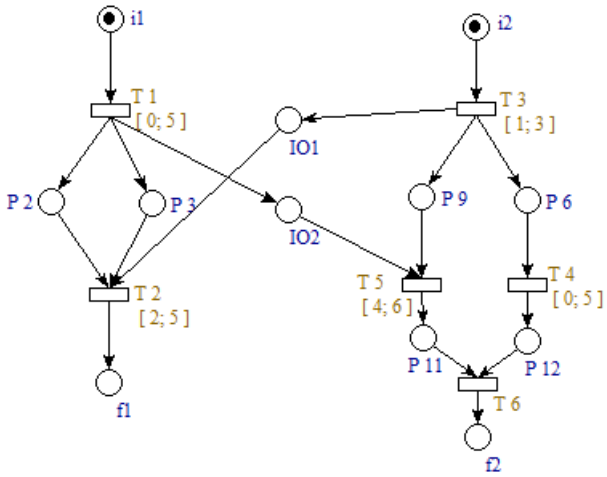

Figure 4: *Example 2 of ToWF-nets composition*

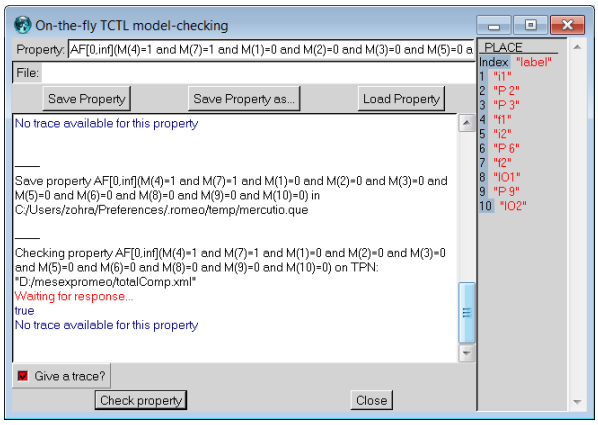

Figure 5: *Total compatibility is verified*

tokens in the beginning of each process and these tokens will be used and released after their use, i.e. at the end of execution, the resource places will regain their same initial markings.

For this, we define the time open WF-nets with shared resources (ToWFR-nets) which stand for a

class of Petri nets dedicated to model workflow processes with: time delays, resource places as well as interface places.

Definition 5 ***(ToWFR-net)***

A ToWFR-net N *is a tuple* $(P, RP, T, F, RF, FI, I, O, W_{RP}, M_0)$ *with:*

- (P, T, F, FI, I, O) *is a ToWF-net,*
- RP *is a set of resource places,*
- I, O, P *and* RP *are disjoint,*
- $RF \subseteq (RP \times T) \cup (T \times RP)$ *is the flow relation for resources,*
- $W_{RP} : RF \mapsto \mathbb{N}$ *is the weight function defining the weight of arcs linking the resource places. To ensure the use of resource places, we require:* $W_{RP}(u) \geq 1 \forall u \in RF$
- M_0 *is the initial marking:* $M_0 = k.i + M_{RP}$ *where* k *is the number of tokens in the initial place* i *and* M_{RP} *is the marking of resource places.*

For the interaction of ToWFR-nets, we use only interface places; resource places are used to model resources sharing between activities of the same process.

In the sequel, we will use these notations:

- IO_j refers to interface place number j
- RP_j refers to resource place number j.

As an example of ToWFR-nets composition, we study a manufacturing lane of three machines which collaborate together to manufacture some product. The composite net describing the manufacturing workflow is given in figure 6.

In this example, each machine will be launched twice (2 tokens in each initial place i_j) and share an internal resource modeled by RP_j with j the number of machines. The three machines communicate via the two interface places IO_1 and IO_2.

Now, to verify if the interacting ToWF-nets communicate correctly and if their collaboration process is sound, we propose to extend the above classes of compatibility properties with the consideration of the concurrent instances ready for execution.

4.1. K-compatibility analysis

We define the following formulas: the k-partial compatibility and the k-total compatibility.

k-partial compatibility

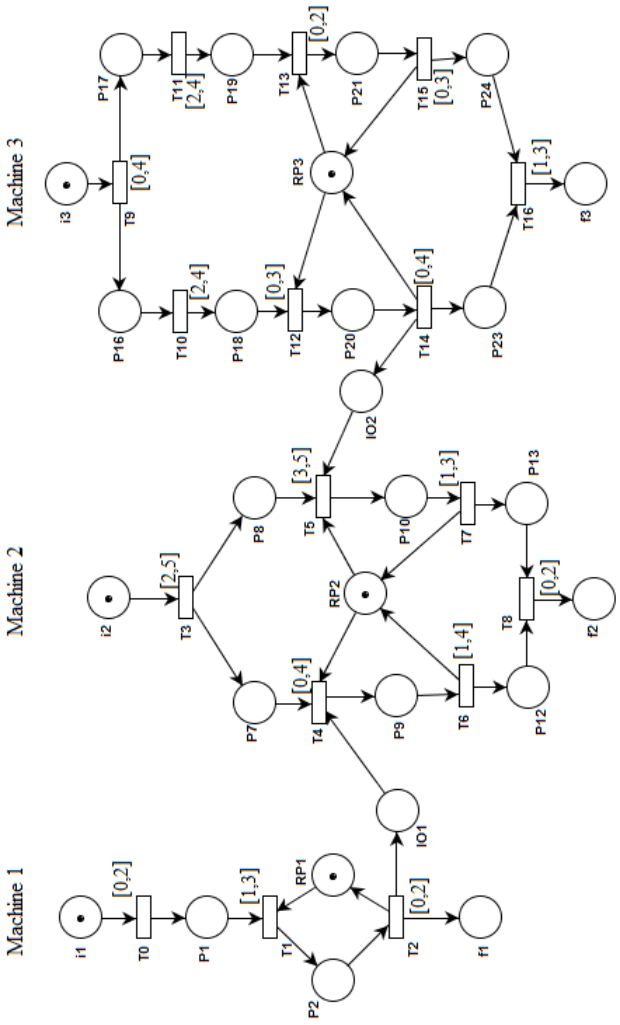

Figure 6: *Composition of three machines' processes*

The k-partial compatibility refers to verify if all the instances of the involved processes terminate without encountering a deadlock. This is possible by extending the deadlock-freeness specified in the previous section with a test of the termination of the k instances. This can be specified by the following TCTL formula:

$$AG[0, \infty[(not(\overset{nbX}{\underset{s=1}{\wedge}}(M(index\_of\_f_s) >= k)) \ => \ not\ deadlock)$$

Where nbX is the number of involved processes

For the three machines example (6) this formula is written in Romeo as follows:

$$AG[0, inf](not(M(4) >= 2\ and\ M(7) >= 2\ and\ M(26) >= 2) \ => \ not\ deadlock)$$

Where 4,7 and 26 are the indexes attributed by Romeo to the three final places.

The figure 7 shows that the two-partial compatibility is guaranteed for the three machines example.

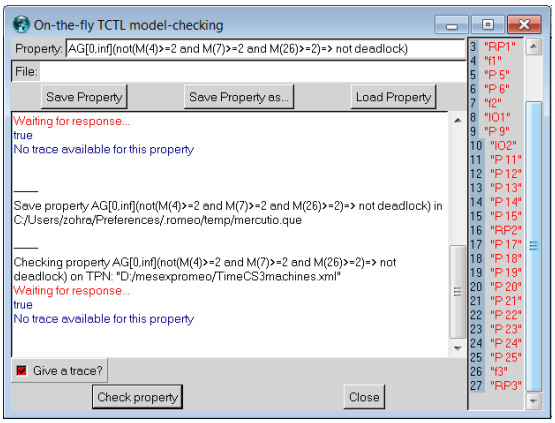

Figure 7: *Analysis example of the two-partial compatibility*

k-total compatibility

The k-total compatibility refers to verify if all the instances of the involved processes terminate properly. This means that we have to verify if we reach the state in which final places have k tokens and resource places regain exactly their initial marking and all the other places are empty. This can be specified by the following TCTL formula:

$$AF[0,\infty[(\overset{nbX}{\underset{s=1}{\wedge}}(M(index\_of\_f_s) = k) \wedge \overset{nbR}{\underset{j=1}{\wedge}}(M(index\_of\_RP_j) = m_{RP_j}) \wedge \overset{nbop}{\underset{j=1}{\wedge}}(M(index\_of\_P_j) = 0))$$

Where nbX refers to the number of interacting processes, nbR refers to the number of resource places, m_{RP_j} refers to the initial marking of resource place RP_j and $nbop$ refers to the number of places which are neither final places nor resource places.

For the three machines example (6) this formula is written in Romeo as follows:

$$AF[0,inf](M(4) = 1\ and\ M(7) = 1\ and\ M(26) = 1\ and\ M(27) = 1\ and\ M(16) = 1\ and\ M(3) = 1\ and\ M(1) =\ and\ M(2) = 0\ and\ M(5) = 0\ and\ M(6) = 0\ and\ M(8) = 0\ and\ M(9) = 0\ and\ M(10) = 0\ and\ M(11) = 0\ and\ M(12) = 0\ and\ M(13) = 0\ and\ M(14) = 0\ and\ M(15) = 0\ and\ M(17) = 0\ and\ M(18) = 0\ and\ M(19) = 0\ and\ M(20) = 0\ and\ M(21) = 0\ and\ M(22) = 0\ and\ M(23) = 0\ and\ M(24) = 0\ and\ M(25) = 0)$$

The figure 8 shows that the two-total compatibility is guaranteed for the three machines example.

After these tests, we highlight some results on the characterization of quantitative properties of

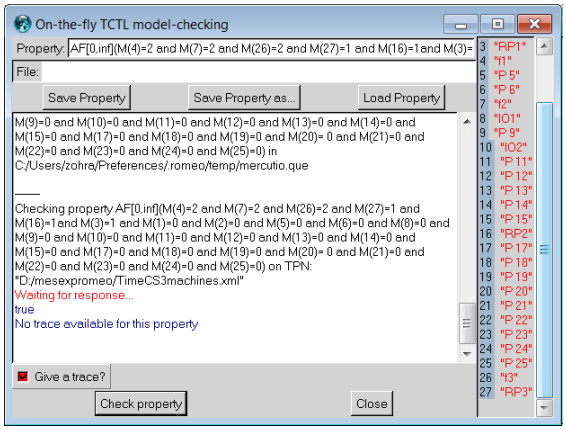

Figure 8: *Analysis example of the two-total compatibility*

ToWF-nets composition. Let us consider $\mathcal{N}$ the time workflow net obtained by composing the involved ToWF-nets and the adding of two special places p_{start} and p_{end} and two transitions t_{start} and t_{end} which connect respectively p_{start} to the the input places of the different ToWF-nets and the different final places to p_{end}.

We can easily prove that if the involved ToWF-nets are totally compatible then $\mathcal{N}$ is sound. In addition if the ToWF-nets are k-totally compatible then $\mathcal{N}$ is k-sound. But the reverse is incorrect, i.e. $\mathcal{N}$ is k-sound $\nrightarrow$ the ToWF-nets are k-totally compatible.

5. RELATED WORK

Several works dealt with the analysis of compatibility properties of processes modeled by open workflow nets or by other formalisms. Wil M. P. van der Aalst and al. (van der Aalst et al. (1998)) showed that two or more processes are compatible if their interfaces are compatible and there are no deadlocks. In addition, other concepts were formulated in relation with compatibility such as strategy and controllability.

Marlon Dumas and al. (Dumas et al. (2008)) studied the incompatibility of Web services and classified it into two types such that the incompatibility of signatures and the protocol incompatibility. The former occurs when a service requests an operation which is not possible from another service. The latter occurs when a service enters in a series of interactions with an other service, but there is no compatibility in the two services orders.

Lucas Bordeaux and al. (Bordeaux et al. (2005)) tackled the verification of Web services compatibility while assuming that not only the exchanged messages are semantically of the same type but also they have the same name. For the modeling of Web services, their work is based on labeled transition systems (LTS). The authors defined three

types of compatibility: absence of deadlock, opposite behavior and unspecified reception.

Wei Tan and al. (Tan et al. (2009)) proposed an approach that checks interface compatibility of Web services described by BPEL, and corrects these services if they are not compatible. To do this, they modeled the composition by SWF-nets, a subclass of CPN (Colored Petri Nets). Then they checked the compatibility of interfaces.

While these approaches dealt with non time processes, we focused on those constrained by time information. To check the compatibility of interacting processes, we chose to extend oWF-nets with delays associated to activities. In addition, we were based on the formal verification in our approach. We mainly used the model checking formal method to check the compatibility classes of ToWF-nets, which shows a counter example in case a property is violated allowing thus to recognize and correct the eventual errors as early as possible.

6. CONCLUSION

Nowadays, technological progress plays a fundamental role in the optimization of production processes in different sectors, especially facing the varieties produced and the constraints of productivity, capability, quality and competitiveness. For this, a prior verification of such processes and their interaction is tremendous.

We proposed first in this paper a model for processes interaction based on Petri nets. Then, we studied the compatibility of these processes by model checking techniques. In particular, we applied a TCTL model checking of these properties and simulated some examples on the Romeo model checker. Finally, we enhanced these results by taking into account several instances ready for execution as well as a possibility of sharing resources. Hence, we introduced to parametric verification of interacting processes. In future, we propose to strengthen this parametric analysis of ToWF-nets and ToWFR-nets.

REFERENCES

Alur, R. and Courchoubetis, C. and Dill, D. (1993) *Model checking in dense real time*. Information and computation (104). pp 2-34.

Atluri, V. and Huang, W. (1996) *An authorization model for workflows*. Proceedings of the 4th European Symposium on Research in Computer Security, London, Springer-Verlag. pp 44-64.

Barkaoui, K. and Ben Ayed, R. and Sbaï, Z. (2007) *Workflow Soundness Verification based on Structure Theory of Petri Nets*. International Journal of Computing and Information Sciences (IJCIS), pp. 51-61.

Berthomieu, B. and Diaz, M. (1991) *Modeling and verification of time dependent systems using time Petri nets*. IEEE Transactions on Software Engineering, 17(3).

Bordeaux, L. and Salaun, G. and Berardi, D. and Mecella, M. (2005) *When are two web services compatible ?*. Sapienza University, 3324.

Boucheneb, H. and Barkaoui, K. (2013) *Reducing interleaving semantics redundancy in reachability analysis of time Petri nets*. ACM Transactions in Embedded Computing Systems (TECS), 12(1), pp 1-24.

Boucheneb, H. and Barkaoui, K. (2012) *Parametric Verification of Time Workflow Nets*. 24th International Conference on Software Engineering (SEKE), pp 375-380.

Berthomieu B. and F. Vernadat F. (2003) *State class constructions for branching analysis of time Petri nets*. TACAS 2003, volume 2619 of Lecture Notes in Computer Science, pp 442-457.

Dumas, M. and Benatallah, B. and Motahari Nezhad, H. (2008) *Web Service Protocols : Compatibility and Adaptation*. Institute of Electrical and Electronics Engineers.

Guermouche, N. and Perrin, O. and Ringeissen, C. (2008) *Timed Specification For Web Services Compatibility Analysis*. Theoretical Computer Science.

Hadjidj, R. and Boucheneb, H. (2009) *On-the-Fly TCTL Model-Checking for Time Petri Nets*. Theoretical Computer Science, 410(42), pp 4241-4261.

Camerzan, I. (2007) *On Soundness for Time Workflow Nets*. Computer Science Journal of Moldova, 15(1), pp 74-87.

De Michelis, G. and Ellis, C. and Memmi, G. (1994) *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms, Zaragoza, Spain..*

Foster, H. and Uchitel, S. and Magee, J. and Kramer, J. (2004) *Compatibility Verification for Web Service Choreography*. Proceedings of IEEE International Conference on Web Services, pp 738-741.

Ellis, C. and Keddara, K. and Rozenberg, G. (1995) *Dynamic change within workflow systems*. Proceedings of conference on Organizational computing systems, pp 10-21.

Esparza, Javier and Silva, Manuel (1989) *Circuits, handles, bridges and nets*. Applications and Theory of Petri Nets, Lecture Notes in Computer Science, 483, pp 210-242.

Martens, A. (2003) *On Compatibility of Web Services*. Petri Net Newsletter, pp 12-20.

Gardey, G. and Lime, D. and Magnin, M. and Roux, O.H. (2005) *Romeo: A Tool for Time Petri Nets Analysis*. Proceeding of 17th International Conference on Computer Aided Verfication (CAV'05), volume 3576 of Lecture Notes in Computer Science, pp 418-423.

Gou, H. and Huang, B. and Liu, W. and Li, Y. and Ren, S. (2001) *Modeling distributed business processes of virtual enterprises based on the object-oriented approach and petri nets*. Systems Man and Cybernetics.

Konur, S. and Fisher, M. and Schewe, S. (2013) *Combined model checking for temporal, probabilistic, and real-time logics*. Theoretical Computer Science, 503, pp 61-88.

Ling, S. and Schmidt, H. (2000) *Time Petri Nets for Workflow Modelling and Analysis*. IEEE International Conference on Systems, Man, and Cybernetics, pp 3039-3044.

Martens, A. (2005) *Analyzing Web service based business processes*. Proceeding of International Conference on Fundamental Approaches to Software Engineering, Part of the European Joint Conferences on Theory and Practice of Software, Lecture Notes in Computer Science vol. 3442.

Massuthe P. and Reisig W. and Schmidt K. (2005) *An Operating Guideline Approach to the SOA*. Annals of Mathematics, Computing and Teleinformatics, pp 35-43.

Ridi, L. and Torrini, J. and Vicario, E. (2012) *Developing a Scheduler with Difference-Bound Matrices and the Floyd-Warshall Algorithm*. IEEE SOFTWARE.

Sbaï, Z. and Barkaoui, K. (2013) *Vérification formelle des processus workflow - Extension aux workflows inter-organisationnels*. Revue Ingénierie des Systèmes d'Information: Ingénierie des systèmes collaboratifs, 18(5), pp 33-57.

Sbaï, Z. and Barkaoui, K. (2012) *Vérification Formelle des Processus Workflow Collaboratifs*. Actes de la conférence francophone sur les Systèmes Collaboratifs (SysCo'12), pp. 197-210.

Boucheneb, H. and Gardey, G. and Roux, O.H. (2009) *TCTL model-checking of Time Petri Nets*. Journal of Logic and Computation, 19(6), pp 1509-1540.

Tan, Wei and Fan, Yushun and Zhou, MengChu (2009) *A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language*. IEEE T. Automation Science and Engineering, 6(1), pp 94-106.

Yoneda, T. and Ryuba, H. (1998) *CTL model checking of time Petri nets using geometric regions*. IEICE Transactions on Information and Systems, pp 297-396.

van der Aalst, W.M.P. and Arjan, J.M. and Christian, S. and Wolf, K. (1998) *Service Interaction: Patterns, Formalization, and Analysis*. 9th Internatinal School on Formal Methods for the design of Computer, Communication and Software Systems.

van der Aalst, W.M.P. (1997) *Verification of Workflow nets*. ICATPN 97, LNCS, 1248.

van der Aalst, W.M.P. (1996) *Three good reasons for using a petri-net-based workflow management system*. International Working Conference on Information and Process Integration in Enterprises (IPIC96), pp 179-201.

van der Aalst, W.M.P. (1993) *Interval timed coloured petri nets and their analysis*. Proceedings of the 14th International Conference on Application and Theory of Petri Nets, London, Springer-Verlag, pp 453-472.

Karsten and Schmidt (2005) *Controllability of open workflow nets*. EMISA. LNI, Bonner Köllen Verlag, pp 236-249.

Sbaï. Z. and Kamel Barkaoui. and Hanifa Boucheneb. (2014) *Compatibility Analysis of Time Open Workflow Nets*. Proceedings of the International Workshop on Petri Nets and Software Engineering, co-located with 35th International Conference on Application and Theory of Petri Nets and Concurrency (PetriNets 2014) and 14th International Conference on Application of Concurrency to System Design (ACSD), pp 249-268.

Sbaï. Z. and Kamel Barkaoui. (2014) *On Compatibility Analysis of Inter Organizational Business Processes*. Enterprise and Organizational Modeling and Simulation - 10th International Workshop, EOMAS 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014, Selected Papers, pp 171-186.