ISSN 2278-3091
Volume 10, No.2, March - April 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse381022021.pdf
https://doi.org/10.30534/ijatcse/2021/381022021

Formal Verification of Control and Operational Behaviors in Industrial Internet of Things Services Composition

Zohra Sbaï
Computer Science Department, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Saudi Arabia
National Engineering School of Tunis, Tunis El Manar University, Tunisia
Email: z.sbai@psau.edu.sa

ABSTRACT

Industrial Internet of Things (IIoT) services composition rely on composing existing IoT services in industrial context in order to obtain an overall service with added value. The composite service's behavior is extremely influenced by the time, availability, and accuracy of the unitary services. Thus, it is extremely important to guarantee a provision of IIoT services as expected during the modeling phase. For this, we lie on formal verification to check all the possible scenarios before to pass through IIoT services provision. We propose first to model each process involved in the composition by open Time Petri Nets. These nets offer interface places for the purpose of process communication with the other processes. Then the composition of open Time Petri Nets is guaranteed via superimposing the interface places and thus obtaining a Time Petri Net modeling the composite process. Finally, control and operational behaviors of IIoT services composition formally checked after being specified in an expressive fragment of TCTL temporal logic.

Key words: IIoT, services composition, Time Petri Nets, model checking.

1. INTRODUCTION

IoT services form a novel yet important concept generated by the growing need of smart objects facilitating daily lives. They may represent customized, pervasive and cyber physical services that are in general designed according to static environment and typical IoT entities that interact together. Thus, in these solutions, services provisioning is measured according to specific user need, current position, sensors, etc.

The challenge now is to guarantee a dynamic provision of IoT services since these latter may run and stop according to a certain space and time under certain conditions. So to ensure their reliability, IoT services need to be designed with considerations of self-configuration and self-optimization. This will so far help to reduce the smart objects damage and to enlarge their life durations and to gain a maximum productivity.

Such issues are even more important in case of industrial internet of things (IIoT). One major issue is that it is extremely important to guarantee a provision of IIoT services as expected during the modeling phase. For this, we lie on formal verification to check all the possible scenarios before to pass through IIoT services provision.

IIoT services composition rely on composing existing IoT services in industrial context in order to obtain an overall service with added value to the end user. The composite service's behavior is extremely influenced by the time, availability, and accuracy of the unitary services. For example, in a manufacturing process, it is important to monitor the production lines starting from the treatment of raw material to the packaging, transport, and tracking of the final products. As another example, in emergency medical care, it is extremely important to guarantee the accuracy and the respect of constraints in the processes involved in the treating of an emergency case for example such as a person who had an accident or a patient in a hospital whose state is aggravated.

In these contexts, as well as more other sophisticated contexts, the composition of existing IIoT services may be guaranteed via the interaction of different services. This interaction may be assured by allowing communication between the different partners of the composition. Communication here may cover messages exchange or even resources sharing such as IoT devices, robots, computers, etc.

Since many partners are invited to communicate together and to operate collectively in order to achieve common goals, a sound control and operational behavior of the overall process should be ensured. An operational behavior refers to a

specific partner's behavior referring to its business logic. The general behavior of any process in the composition is described in terms of control behavior. Considering the importance and the actual need of the IIoT services compositions, the challenge here is to verify the aforementioned types of behaviors in an early stage.

In order to be able to reveal the possible errors and correct them in the earliest, we propose for the IIoT system to be modeled by Time Petri nets and formally verified by model checking [12-15,33]. More precisely, we propose to model each process involved in the composition by an open Time Petri Net, characterized by offering interface places for the purpose of communication with the other involved processes. Then the composition of open Time Petri Nets is guaranteed via superimposing interface places and thus obtaining a Time Petri Net modeling the overall (composite) process. This concept of using a variant of TPN with input/output interfaces is first used in [18-20,32] to model communicating workflow processes.

The remainder of this paper consists of five sections. Section two presents a motivation to this research. Section three proposes a Time Petri Nets based model of IIoT services composition. After that, model checking of the IIoT services composition is discussed in section four. The last section concludes this research and exposes new work directions.

2. RESEARCH MOTIVATION

The Internet of Things (IoT) has quickly been risen lately. The IoT depends on keen and interconnected items in a dynamic and worldwide system foundation. It is for the most part described by little certifiable articles, circulated generally, with constrained capacity and preparing limit, however fit for estimating, deriving, understanding, and in any event, adjusting their condition. It doesn't just concern complex gadgets, for example, cell phones, yet in addition straightforward regular objects, for example, watches, indoor regulators, apparel, and so on [22, 23]. These items, going about as sensors or actuators, can associate with one another, hence impacting definitely the everyday lives of likely clients. For these reasons, IoT is more and more playing a decisive role in all the domains such as healthcare, intelligent transportation, home automation, assisted living, farming, and automated manufacturing. Based on these considerations, the United States National Intelligence Council has stated that the IoT is one of six technologies that will have potential impact on American interests by 2025 [23].

In IoT, sensors and actuators monitor the state of their environment, i.e. obtain information on temperature, movement, position, etc. They constitute a network generally composed of a potentially large number of nodes. These sensors have to face many communication problems such as security and confidentiality, mobility, reliability, robustness, scalability and especially the availability of their resources (energy, storage and processing capacity, bandwidth, etc.).
From a user perspective, the services gained from these sensors are those which matter essentially, especially when many services have to be fulfilled together to meet the user's need. For that, many platforms are being implemented in the context of IoT (IoT6 [24], BigClout [25], AWS IoT [26], BlueMix [27], etc.). However, these platforms have in general implicit management aspects, adapted to a particular context. The architect should be able to select, personalize and adapt his solutions according to behavioral evolution. The proposed architectures do not offer a loose coupling of the components allowing a re-composition, according to the behavioral changes.

In this context, we are motivated by tackling the problem of services composition in IoT and their formal verification that helps to diagnose the composition and detect the possible errors in an early time. On the one hand, this will help to be sure of the absence of errors in a composition of IoT services and on the other hand, in case of errors' detection at an earlier time, re-composition with other compatible services is feasible.

We choose to apply our approach in the IIoT domain, which essentially consider the application of IoT technologies in the sector of industry. It represents a form of the rapid evolution of digital transformation. It concerns all sectors, from energy production to agriculture and municipal management, allowing thus to create the industry 4.0. IIoT makes factories smarter and more efficient since each link in the production chain will be in constant communication to produce faster and more efficiently. The three typical applications where IIoT demonstrates its benefits are:

- Production: a means of production that is both connected and intelligent, which generates considerable amounts of data, the analysis of which largely determines the ability of an organization to improve its operation and remain competitive.
- Supply chain: thanks to sensor-managed stock, IIoT technology can place orders for supplies just before the stock runs out. Significant time savings allowing employees to focus on other tasks may be guaranteed.
- Healthcare: the IIoT allows the healthcare industry to be more efficient and responsive with devices that monitor patients remotely and alert the medical team as soon as a change in the patient's condition happens.

We can conclude that the strengths of IIoT technology are mainly based on the fact that it promotes constant communication between all the elements, and it creates a distributed permanent intelligence that will contribute to a socially benevolent and more ecological future. Thus, guaranteeing a well-controlled IIoT will guarantee an ultra-connected and smarter industry.

The IIoT major focus is on interoperability between devices and machines and the transfer and control of information and responses. It strongly depends on M2M (Machine to Machine) communications. For this, it is highly important to efficiently ensure the control of IIoT.

Given that formal methods are very promoting, we propose to rely on them in this work. We focus especially on model checking techniques [7,8,9,17] and propose a complete approach to specify and verify services composition in IIoT. Model checking is based on specifying the model as well as the behavior to be verified by means of techniques dedicated to the model checker that will be used. For this, we choose to lie on Petri nets [21] and this choice is explained by their well expressiveness and power of modeling discrete event systems. More precisely we adopt open Time Petri Nets which allow us in the first hand to consider the timing constraints if they exist such as the delays of tasks' executions, and in the second hand to use interfaces guaranteeing the communications between the involved IIoT services.

3. OPEN TIME PETRI NETS FOR IIOT SERVICES MODELING

Petri nets form a powerful modeling and analysis tool of Discrete Event Systems. They have proven their efficient application in a large spectrum of services going from manufacturing, healthcare, traffic, to communications, and software systems. Basically, a Petri net consists of a directed graph composed of transitions and places connected via arcs. Bars or rectangles are used to schematize the transitions which refer to the events that may occur. The places (drawn as circles) represent the conditions under which events may occur. These two types of nodes are connected by arrows specifying the places that are preconditions and/or postconditions for the transitions. This graphical notation makes it easier to describe choices, iterations, and concurrent executions. Beyond this graphical simple notation, a mathematical theory is well-developed for Petri nets to define their execution semantics.

The execution of the aforementioned events specified by the transitions is guaranteed by their firing. Before its firing, a transition has to be enabled (also said to be firable) i.e. its precondition is satisfied. This condition refers to the presence of sufficient marks (known as tokens) associated with each transition's input. Once the transition fires, those tokens will be consumed (removed) but other tokens will be created in each of the transition's output places. The number of tokens to be consumed or produced may be specified on the arcs. When no number is indicated, it is considered that only one token is to be consumed or produced.

At a given moment, the number of marks in each of the net places is known to be a marking. The firing of a transition causes a new marking to be reachable, resulting thus to new transitions to be enabled. The overall possible tokens' distributions define the state space of the Petri net. When many transitions may fire from the same marking, they will occur in any order, hence the firing process is nondeterministic. Given the nondeterministic firing process and the count of marks in the Petri net, the potential of Petri nets in modeling distributed systems and controlling their concurrent behaviors is proven.

Petri nets have been widely studied and extended to subclasses to deal with complex constraints. For instance many variants treat the timing information in the modelled events, others treat the communication over processes. To deal with the timing information, we propose to adopt Time Petri Nets, and for the communication inter processes, we refer to open Petri nets. In this work, the two Petri nets extensions mentioned above are combined to obtain a new subclass baptised open Time Petri Nets (oTPN).

When we augment the transitions with intervals of time, we obtain a Time Petri net [1-5]. Each interval specifies the minimum time to fire the transition (if enabled) as well as its last time to fire.

Open Petri nets (or simply open nets) are used to model processes communicating together via input/output places attached to each process. These places are used as interfaces to communicate with other processes. This communication may be seen as a sending and a reception of messages between all the processes forming the overall composition.
We combined these two subclasses to define open Time Petri Nets *(oTPN).*

Definition 1. *An* open Time Petri Nets *is defined by the tuple* $(P, T, I, O, {}^\bullet(.), (.)^\bullet, \alpha, \beta, M_0)$ *with:*

- *P*: bounded set of places,
- *T*: bounded set of transitions,
- $P \cap T = \emptyset$,
- *I*: bounded set of interface places serving as inputs: ${}^\bullet I = \emptyset$,
- *O*: bounded set of interface places serving as outputs: $O^\bullet = \emptyset$,
- *P, I,* and *O* are disjoint,
- ${}^\bullet(.) : T \to N^{P \cup I \cup O}$ *defines the markings of the pre-places of the transitions,*
- $(.)^\bullet : T \to N^{P \cup I \cup O}$ *defines the markings of the post-places of the transitions,*
- M_0 denotes the initial marks' distribution on the places ($M_0 \in N^{P \cup I \cup O}$),
- $\alpha : T \to Q^+$ and $\beta : T \to Q^+ \cup \{\infty\}$ define the minimum and maximum firing time.

To illustrate the different aforementioned concepts, let us consider a case study. In IIoT context, a large number of machines and devices will communicate in a smart way in

order to achieve a common goal. To guarantee an overall optimal execution, we propose to model each service provided by one of these partners in terms of an oTPN and we define the communications of them via interfaces.

Figure 1 illustrates the modeling of three IIoT services by three oTPNs in which the input and output interfaces are precised by places I_{ij} and O_{ij} where *i* and *j* refer to services *i* and *j*. For sake of simplicity and focus on the interfaces, the timing intervals are omitted from the transitions.

It is mentioned via the gray circles that each interface place which is an input to an IIoT service has a relation with another interface place standing as an output to an IIoT service. For instance, the first output interface of service 1 (O_{11}) designs a prerequisite to tasks from both service 2 and service 3. Thus I_{21} and I_{32} model the input interfaces of respectively service 2 and service 3 that would ensure the prerequisite satisfiability.
The overall composition of the different oTPNs is obtained by superimposing the interfaces of the different partners. The input interfaces of partner 1 may be output interfaces belonging to other partners and vice versa. For the example of Figure 1, I_{11} and O_{21} will be considered as one interface place assuring a communication between service 1 and service 2. O_{11}, I_{21}, and I_{32} will be superimposed in only one interface, and same for the other communications between services 1 and 3 and services 2 and 3.

In general, the composition of more than two oTPNs will result in a Timed Petri Net according to definition 2.

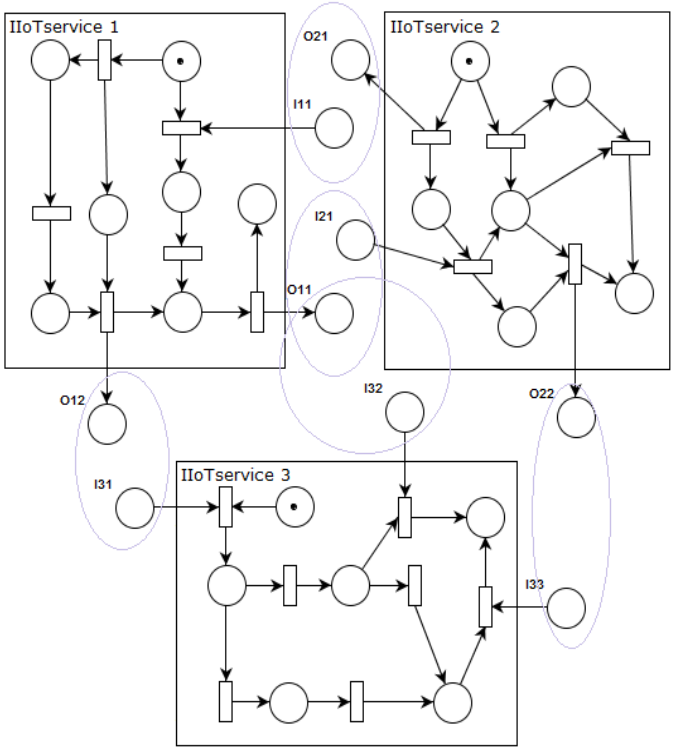

Figure 1: Modeling of three IIoT services by three oTPNs

Definition 2. *The composition of X oTPNs* $N_1, .., N_X$ *is a tuple* $(P, T, {}^\bullet(.), (.)^\bullet, \alpha, \beta, M_0)$ *such that:*

- $P = \cup_{i=1}^{X} \quad P_i \cup \cup_{i=1}^{X} \quad I_i \cup \cup_{i=1}^{X} \quad O_i$
- $T = \cup_{i=1}^{X} \quad T_i$
- ${}^\bullet(.) : T \to N^P$ and $(.)^\bullet : T \to N^P$ *define the pre (post) markings related to transitions firings,*
- M_0 *is the initial tokens' distribution* ($M_0 \in N^P$),
- $\alpha: T \to Q^+$ *defines the minimal time of transitions' firing,*
- $\beta: T \to Q^+ \cup \{\infty\}$ *defines the maximum time of transition's firing.*

For each transition t_i, α_i and β_i are denoted resp. $\alpha(t_i)$ and $\beta(t_i)$.

For a time Petri net N, for every place p, $M(p)$ denotes the marks count in p. M denotes the marks' distribution over the net.

A transition t_i is said to be firable from M if $M \geq {}^\bullet t_i$. All the transitions that can be fired form M form a set called $Enabled\,(M)$.

Once a transition t_i is fired, we are interested in the newly firable transitions. A transition t_k is newly enabled if and only if $M - {}^\bullet t_i$ does not enable the transition t_k and $M + t_i{}^\bullet - {}^\bullet t_i$ allows t_k to be enabled. In the same manner, once t_i is fired, we regroup all the transitions newly firable in a set called $Enabled(M, t_i)$.

Definition 3. The semantics S_N of a time Petri net N is defined by the quadruplet $(Q, \{q_0\}, \Sigma, \to)$ where*:*

- $Q = N^P \times (R_{\geq 0})^T$,
- $q_0 = (M_0, \bar{0})$,
- $\Sigma = T$,
- $\to \in Q \times (T \cup R_{\geq 0}) \times Q$ is a relation of transitions that may be discrete or continuous.

- Discrete transition: $\forall t \in T$

$$(M,v)\, t_j \to (M',v') \; iff \; \{M \geq {}^\bullet t_j \;\; M' = M - {}^\bullet t_j + t_j{}^\bullet \;\; \alpha_j \leq v_j \leq \beta_j \; \forall c \in [1..n], \;\; v'_c = \begin{cases} 0 \; if \; t_c \in Enabled(M, t_j) \\ v_c \; otherwise \end{cases}$$

- Continuous transition: $\forall r \in R_{\geq 0}$ *:*

$$(M,v)\, r \to (M,v') \; iff \; \{v' = v + r \; \forall c \in [1..n], M \geq {}^\bullet t_c \Rightarrow v'_c \leq \beta_c$$

All the possible marks' distributions of a TPN define its state space. If this set is bounded then the net is bounded. To delimit the state space of the TPN, every state that is reachable by $Enabled(M, t)$ for all M and t has to be computed.

This state space may be considered to check important properties of the overall composition of IIoT services. We present in the following section a model checking based approach of verifying quantitative and qualitative properties of IIoT services composition.

4. $TCTL_h^{\Delta}$ TO VERIFY IIOT SERVICES COMPOSITION

With regard to the industrial revolution and the emergence of industrial Internet of Things, several research challenges [28-31] are actually being addressed. Among these challenges, we believe that it is extremely important to detect and correct at the earliest the possible errors that may occur during the running of the IIoT based systems. For this purpose model checking is a promising method that permits to count the attainable states of the net automatically and to verify if it meets its specification. To achieve this goal the system as well as the behavior to be checked should be specified in formal models according to the used tool of model checking.

Let us consider three machines in a smart factory delivering specific and critical services while sharing the same resources or needing an input from the same side. Suppose that these three machines perform each of which some tasks in specific amounts of time. This situation may be illustrated by the TPN of the Figure 2. The place SR represents the shared resource referring to an IoT device for example.

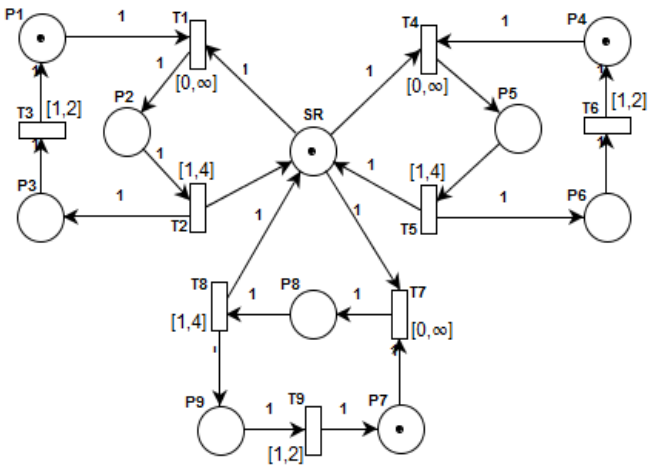

Figure 2:.Modeling 3 communicating machines

Table 1 explains the tasks modeled by the transitions T1,..,T6 in figure 2.

Table 1: Specification of transitions in figure 2

Transition	Indication
T1	Machine 1 blocks the necessary resources and preprocesses for the main task
T2	Machine 1 achieves its main task and releases the shared resource
T3	Machine 1 is being ready to reprocess the main task
T4	Machine 2 blocks the necessary resources and preprocesses for the main task
T5	Machine 2 achieves its main task and releases the shared resource
T6	Machine 2 is being ready to reprocess the main task
T7	Machine 3 blocks the necessary resources and preprocesses for the main task
T8	Machine 4 achieves its main task and releases the shared resource
T9	Machine 5 is being ready to reprocess the main task

The place SR represents a resource shared between the three machines, it is tremendous for the main processing of the machines, and each machine has to release it after finishing its main processing. Suppose we have a constraint on the use of this shared resource: there should be a certain amount of time not using this resource interleaving its use by the different machines.

So, according to the timing constraints defined in the TPN of Figure 2, we have to verify the following: during any interval of a duration not exceeding four time units, machines 2 and 3 are not allowed to begin pre-processing during at least 2 units of time after the first machine finishes processing its main job.
This type of property may be specified in a temporal logic expressing the delays as well as the durations. In [15], the timed temporal logic TCTL is extended with new techniques of considering the activation delays and the durations. As a result, the authors defined $TCTL_h^{\Delta}$ as a new timed temporal logic with new modalities of delays and durations combined from $TCTL^{\Delta}$ and $TCTL_h$.

$TCTL_h^{\Delta}$ based checking of the composite service depends on the state space computation. Here we consider on the fly techniques [6,10] of $TCTL_h^{\Delta}$ model checking which permit to get rid of the explosion of the number of states. Actually, the states are dynamically computed while the algorithm is

executing; which permits not only to save time and space but also to show the result of the verification at the earliest.

This $TCTL_h^\Delta$ model-checking may be assured via the verification of $TPN - TCTL_h^\Delta$ properties implemented on Romeo model checker [16] as an extension of TCTL model checking of TPN [11].

Definition 4. $TPN - TCTL_h^\Delta$ *(*$TCTL_h^\Delta$ *for TPN) is defined as follows:*

$$TPN - TCTL_h^\Delta ::= GMEC_h^\Delta \mid false \mid \neg\phi \mid \phi \Rightarrow \psi \mid \exists\phi\, U^k\psi \mid \forall\phi\, U^k\psi$$

where: $false$ *is a keyword and* $k \in N$,

$$GMEC_h^\Delta = \text{not p} \mid \text{p and q} \mid \text{p or q} \mid \text{p} => \text{q} \mid \text{aMi} + \text{bMj} \sim \text{k} \mid \text{aMi} - \text{bMj} \sim \text{k} \mid \text{bounded(k)} \mid \text{deadlock} \mid \text{x} \sim \text{c} \mid (\text{x} - \text{y}) \sim \text{c} \mid \text{x in p}$$

with $\sim \in \{<, \leq, >, \geq, =\}$, in *: a freeze quantifier, bounded and deadlock are defined in Romeo.*

So a property expressed in $TPN - TCTL_h^\Delta$ may be defined as follows:

$$E(p)U(q) \mid A(p)U(q) \mid EF(p) \mid AF(p) \mid EG(p) \mid AG(p) \mid (p) \rightarrow (q) \mid E(p)U^k(q) \mid A(p)U^k(q) \mid AF^k(p) \mid EF^k(p) \mid AG^k(p) \mid EG^k(p) \mid (p) \rightarrow^k (q).$$

Where: $p \text{ and } q \in TPN - TCTL_h^\Delta$, U, E, A, and G are the known quantifiers of temporal logics, and $(\neg, \Rightarrow, \wedge, \vee)$ are the ordinary logical operators.

For the example in Figure 2, recall that we want to check that in any interval of a duration not exceeding four time units, machine 2 as well as machine 3 can't begin pre-processing during at least 2 units of time after the first machine finishes processing its main job. This constraint is expressed in $TPN - TCTL_h^\Delta$ by the following formula:

$$AG\,(x\ in\ (A(\ \neg P_5)\ U^4(P_3\ and\ x \leq 2)))$$

In order to verify these properties, we focus on the semantics of $TPN - TCTL_h^\Delta$. Consider a time Petri net $N = (P, T, \bullet(.), (.)\bullet, \alpha, \beta, M_0)$, its semantics $S_N = (Q, \{q_0\}, T, \rightarrow)$ and $\rho = (s_0, v_0)\ (d_1, t_1) \rightarrow \cdots\ (d_n, t_n) \rightarrow\ (s_n, v_n) \in [\![N]\!]$ an execution of NN.

For a state $q = (M, v)$, the following lines define the validity of a formula specified in $TPN - TCTL_h^\Delta$:

- $q \vDash GMEC_h^\Delta$ *iff* $M \vDash GMEC_h^\Delta$,
- $q \nvDash false$,
- $q \vDash \neg\phi$ *iff* $q \nvDash \phi$,
- $q \vDash \phi \Rightarrow \psi$ *iff* $q \nvDash \phi$ *or* $q \vDash \psi$,
- $q \vDash \exists\phi\, U^k\psi$ *iff for some* $\rho \in [\![N]\!]$, *for some* $t \leq k$, $\rho(t) \vDash \psi$, *and for all* $0 \leq t' \leq t$, $\rho(t') \vDash \phi$,
- $q \vDash \forall\phi\, U^k\psi$ *iff for every* $\rho \in [\![N]\!]$, *for some* $t \leq k$, $\rho(t) \vDash \psi$, *and for all* $0 \leq t' \leq t$, $\rho(t') \vDash \phi$.

A time Petri net N satisfies a formula φ specified in $TPN - TCTL_h^\Delta$ (denoted as $N \vDash \varphi$), if and only if Ψ is satisfied by the first state of its semantics S_N.

5. CONCLUSION

Industrial Internet of Things form the main technology allowing the industry 4.0 to emerge. It concerns all sectors, from energy production to agriculture to healthcare systems. IIoT surrounds cyber physical systems, smart factories, industrial robots, actuators, etc. In these systems, assumed to be safe-critical, interactions between different devices and/or machines are very frequent and tremendous as well. For this, we lied on formal verification to verify the control and operational behaviors of communicating services in the context of IIoT.

We proposed a model checking based method to formally analyze IIoT services composition. First, the involved IIoT services are modeled in terms of open Time Petri Nets, leading thus to model the overall composite service in terms of a TPN. Then, the properties of correct behavior and communication between partners are specified in terms of $TCTL_h^\Delta$. On the fly $TCTL_h^\Delta$ model checking may be finally applied on the obtained TPN in Romeo model checker.

While in this paper, we focused on specifying the functional properties assuming time delays between the occurrence of events as well as the tasks durations to be checked, we plan in the future to study more properties related to safety and security. Also, it is important to study the new constraints and properties specific to IIoT such as scalability, usability, and privacy.

REFERENCES

[1] P. Merlin, **A study of the recoverability of computing systems**, Ph.D. thesis, Dep. of Information and Computer Science, University of California, Irvine, CA (1974).

[2] W. Khansa, J. Denat, and S. Collart-Dutilleul, **P-Time Petri Nets for manufacturing systems**, in: International Workshop on Discrete Event Systems, (WODES'96), Edinburgh (U.K.), 1996, pp. 94–102. doi:10.1016/ S1474-6670(17)43571-3.

[3] B. Berthomieu, and M. Diaz, **Modeling and verification of time dependent systems using Time Petri Nets**, IEEE transactions on software engineering 17 (1991) 259–273.

[4] D. de Frutos Escrig, V. Valero-Ruiz, and O. Marroquin-Alonso, **Decidability of properties of timed-arc Petri nets**, in: ICATPN'00, Vol. 1825 of 540 Lecture Notes in Computer Science, 2000, pp. 187–206. doi:10.1007/3-540-44988-4_12.

[5] T. Yoneda, and H. Ryuba, **CTL model checking of Time Petri Nets using geometric regions**, IEEE Transactions on Information and Systems E99– D (3) (1998) 297–396.

[6] O. Kupferman, T. A-Henzinger, and M. Y-Vardi, **A space, T-efficient onthe-fly algorithm for real-time model checking**, in: Proceedings of the Seventh International Conference on Concurrency Theory (CONCUR), Vol. Lecture Notes in Computer Science 1119, 1996, pp. 514–529. doi: 10.1007/3-540-61604-7_73.

[7] F. Laroussinie, **Model checking temporisé: Algorithmes efficaces et complexité**, Master's thesis, ENSCachan, December 2005.

[8] H. Mokadem, B. Berard, P. Bouyer, and F. Laroussinie, **Timed temporal logics for abstracting transient states**, Vol. 4218, 4th International Symposium on Automated Technology for Verification and Analysis (ATVA 2006), 2006, pp. 337–351.

[9] H. Kugler, D. Harel, A. Pnueli, Y. Lu, and Y.Bontemps, **Temporal logic for scenario-based specifications**, in: TACAS'05 Proceedings of the 11th international conference on Tools and Algorithms for the Construction and Analysis of Systems, 2005, pp. 445–460. doi:10.1007/978-3-540-31980-1_29.

[10] R. Hadjidj, and H. Boucheneb, **On-the-fly TCTL model checking for Time Petri Nets,** Theoretical Computer Science 410(42) (2009) 4241–4261. URL https://www.sciencedirect.com /science/ article/pii/S0304397509004125

[11] H. Boucheneb, G. Gardey, and O. H-Roux, **TCTL model checking of Time Petri Nets**, Journal of Logic and Computation 19 (2009) 1509–1540. doi: 10.1093/logcom/exp036.

[12] N. Jbeli and Z. Sbaï, **On Improving Model Checking of Time Petri Nets and Its Application to the Formal Verification**. Int. J. Serv. Sci. Manag. Eng. Technol. 12(4), 2021.

[13] N. Jbeli, Z. Sbai, and R. Ben Ayed. **On expressiveness of TCTL∆h for Model Checking Distributed Systems**, in: International Conference on Computational Collective Intelligence ICCCI 2016, Halkidiki, Greece, September 28-30, Springer 2016, Proceedings, Part I, Lecture Notes in Computer Science 9875, pp. 323–332.

[14] N. Jbeli, Z. Sbai, and R. Ben Ayed, **On the fly model-checking of TPN: $TPN-TCTL_{\frac{1}{2}}$ for quantitative verification of information systems**, in: World Conference on Information Systems and Technologies (WorldCist'18), Naples, Italy, March 27-29, 2018, Vol. 746. Springer, Proceedings Part II, Trends and Advances in Information Systems and Technologies. Advances in Intelligent Systems and Computing, pp. 441–451.

[15] N. Jbeli, Z. Sbai, and R. Ben Ayed, $TCTL_{\frac{1}{2}}$**Model Checking of Time Petri nets**, LNCS Transactions on Computational Collective Intelligence, Vol. 30, pp. 242-262. Lecture Notes in Computer Science 11120, Springer 2018, ISBN 978-3-319-99809-1, 2018.

[16] G. Gardey, D. Lime, M. Magnin, and O. H-Roux. **Romeo: a tool for analyzing Time Petri Netss,** in: 17th International Conference on Computer Aided Verfication (CAV'05), Vol. 3576, 2005, pp. 418–423. URL http://romeo.rts-software.org

[17] K. Y-Rozier, **Linear temporal logic symbolic model checking**, Computer Science Review 5:2 (2011) 163–203. doi:10.1016/j.cosrev.2010.06.002.

[18] Z. Sbaï, and R. Guerfel, **CTL Model Checking of Web Services Composition based on Open Workflow Nets Modeling**. Int. J. Serv. Sci. Manag. Eng. Technol. 7(1): 27-42 (2016).

[19] Z. Sbaï, and K. Barkaoui. **On quantitative Analysis of Time Open Workflow Nets and Parametric Extension**, Proceedings of the 9th Workshop on Verification and Evaluation of Computer and Communication Systems, VECoS 2015, Bucharest, Romania, September 10-11, 2015, pp. 97-108.

[20] Z. Sbaï, K. Barkaoui, and H. Boucheneb. **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 and 14th International Conference on Application of Concurrency to System Design Tunis, Tunisia, June 23-24, 2014, pp. 249-268.

[21] K. Barkaoui, R. Ben Ayed, and Z. Sbaï. **Workflow Soundness Verification based on Structure Theory of Petri Nets**, International Journal of Computing & Information Sciences, ISSN 1708-0460 (Print), ISSN 1708-0479 (On-Line), Vol. 4, No. 1, pp. 51-61, 2007.

[22] Evangelos A. Kosmatos, Nikolaos D. Tselikas, and Anthony C. Boucouvalas. **Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture**, Advances in Internet of Things, 01(01) :5–12, 2011. ISSN 2161-6817, 2161-6825. doi : 10.4236/ait.2011.11002. URL http://www.scirp.org/journal/doi.aspx?DOI=10. 4236/ait.2011.11002.

[23] National Intelligence Council. Disruptive Civil Technologies. Six technologies with potential impacts on US interests out to 2025. 2008.

[24] IoT6. IoT6, 2018. URL https://iot6.eu/.

[25] BigClouT. BigClouT - **Big data meeting Cloud and IoT for empowering the citizen clout in smart cities** - EC funded project, 2018. URL http://bigclout.eu/.

[26] AWS IoT. **AWS IoT**, 2018. URL https://aws.amazon.com/iot/.

[27] IBM Bluemix. **IBM Bluemix, 2018**. URL https://www.ibm.com/cloud-computing/bluemix.

[28] Natarajan, Muthukumar & Srinivasan, Seshadhri & Ramkumar, Kannan & Pal, Deepak & Vain, Juri & Ramaswamy, Srini. (2019). **A model-based approach for design and verification of Industrial Internet of Things**, Future Generation Computer Systems. 95. 10.1016/j.future.2018.12.012.

[29] L. Antão, R. Pinto, J. Reis and G. Gonçalves, **Requirements for Testing and Validating the Industrial Internet of Things**, 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Vasteras, 2018, pp. 110-115.

[30] A. Souri, and M. Norouzi. **A State-of-the-Art Survey on Formal Verification of the Internet of Things Applications**, J Serv Sci Res 11, 47–67 (2019). https://doi.org/10.1007/s12927-019-0003-8.

[31] S. Ahmad, S. Malik, I. Ullah, DH. Park, K. Kim, and D. Kim (2019). **Towards the Design of a Formal Verification and Evaluation Tool of Real-Time Tasks Scheduling of IoT Applications**, Sustainability 11(1): 204.

[32] R. Guerfel, and Z. Sbaï, **D&A4WSC as a Design and Analysis Framework of Web Services Composition**, 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 and 14th International Conference on Application of Concurrency to System Design, Tunis, Tunisia, June 23-24, 2014, pp. 337-338.

[33] A. Chtourou, and Z. Sbaï, **Towards $TCTL_A^A$ model checking of time Petri nets**. International Conference on Control, Decision and Information Technologies, CoDIT 2016, Saint Julian's, Malta, April 6-8, 2016, pp. 563-568.