

Temporal Constraint in Web Service Composition

Bey Fella, Samia Bouyakoub and Abdelkader Belkhir
Institute of Electronics & Computer Sciences, USTHB University, Algiers, Algeria

Keywords: Web Services Composition, Petri Network, Temporal Constraints.

Abstract: Web service composition is studied by many works, and constitutes the heart of a great research activity. However, the majority of this work does not take into account all temporal constraints imposed by the service provider and the users in the composition process. Incorporating temporal constraints in Web service composition result in more complex model and make crucial the verification of temporal consistence during the modeling (at design time) and then during the execution (at run time). In this paper, we presented H-Service-Net model for Web service composition with time constraints, and propose a modular approach for modeling composition with time constraint using Extend time unit system (XTUS), Allen's interval algebra and comparison operators in a time Petri net model.

1 INTRODUCTION

The compositions of Web services allow to determine a combination of services that meets the customer's request, this composition is provided as a single service. Moreover, the composition of Web services should consider the time constraints associated with this composition.

It is possible to identify at least two types of services available on the Web. The former include services that require an immediate response, such as dictionaries or weather services online. This type of service allows the user to get immediate answers. The second type of service refers to Web services that require estimates or more generally a negotiation such as travel agency, building a house or production and delivery of products.

The terms orchestration and choreography describe two aspects of creating business processes from composite Web services. Orchestration refers to an executable business process that can interact with both internal and external Web services, the interactions occur at the message level. They include business logic and task execution order, and they can span application and organizations to define a long-lived, transactional, multi-step process model. Orchestration always represents control from one part's perspective. This differs from choreography, (Peltz, 2003).

This article presents the definition of a formal model that supports the necessary abstractions such

as time constraint, time consistency and proposing a framework that meets the needs of the user with the temporal properties. The content of this article is organized as follows:

Section 2 presents some related work, Section 3 presents our major objectives, Section 4 presents our model named 'H-Service-Net', and it is illustrated with an example. Finally, Section 5 concludes the paper.

2 RELATED WORK

Incorporating the time factor in the process of composing Web services has become essential for the implementation of time-based Web services. Several works have been proposed in the context of modeling time constraints in Web services (Dai, 2007) (CHEN, 2011), (Hamadi, 2003), (Hamadi, 2008). In the next section, we describe the approaches presented in the literature for the development of service composition based on time constraints.

There are several formalisms of modeling Web service composition using Petri nets (Dai, 2007), (CHEN, 2011), (Hamadi, 2003) which is one of the most known formalisms and the most adapted to express the concurrent systems. Hamadi et al. (Hamadi, 2003), (Hamadi, 2008) propose a model based on Petri nets for Web service composition and model the control flow of Web services. Diaz et al.

(Diaz, 2006) use formal methods for describing and analyzing the behavior of Web Services, including time restrictions. Dai et al. (Dai, 2007) present a time constraint modeling and analyzing method for Web processes based on OWL-S annotations. Lallali et al. (Lallali, 2007), (Lallali, 2008) proposed a new formalism called Web service time extended finite state machine (WS-TEFSM) and an intermediate format (IF) which enables modeling of timing constraints in BPEL. Rouached et al. (Rouached, 2006) propose an event driven approach for checking, functional and non-functional consistency of Web service composition expressed in BPEL by using the Event Calculus framework. The paper of Kazhamiakin et al. (Kazhamiakin, 2004) introduced a new formalism called WSTTS (Web service time transition system) which enables verification of time constraints in Web service compositions using model checking for WS-BPEL processes to capture the time behavior. Lastly, Ting-Wei et al. (CHEN, 2011) present a verification approach of time constraint consistency of Web service composition, which transform OWL-S to ETPN (extended time Petri net), an algorithm of time consistency verification in Web service composition is proposed but this paper did not support time constraints for the execution of the service.

The composition of Web services is an issue that has been studied by many researches and is the core to a wide research activity. However, most studies (Diaz, 2006), (Lallali, 2007), (Kazhamiakin, 2004), (Hamadi, 2003), (CHEN, 2011), (Rouached, 2006), (Dai, 2007), (Hamadi, 2008) did not consider all types of temporal constraints "local, global, relative, absolute..." imposed by the composition, the client and the service provider. Therefore, the results did not satisfy the needs and time preferences. Moreover, works like (Hamadi, 2003), (Kazhamiakin, 2004) can only answer whether there is temporal inconsistency, but they did not provide a concrete solution.

3 MAJOR OBJECTIVES

To guaranty vulnerabilities of web service composition and control system behavior is an important criterion and can be done by defining a temporal consistency at design and run time of the composition

Recently, Web service composition model and languages do not allow defining all types of temporal constraint imposed by the client, the

provider and the composition in a declarative and formal way and verify temporal consistency at design and run time, defining if there is a temporal contradiction at design time is important because modification of temporal constraint cannot be done at run time.

4 COMPOSED WEB SERVICES AS TIME PETRI NETS

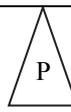
4.1 H-Service-Net: A Temporal Model for Web Service Composition

Although the temporal RDP allows to define temporal performance information, there are not enough for modeling any complex system such as temporal constraints in web service composition. For this, we have extended the timed RDP in order to define a well-suited timed model for the formalization of a web service orchestration. Thus, we have defined an extended time Petri net called H-Service-Net.

The H-Service-Net model (an acronym for Hierarchical Service Net) is a time Petri net-based model. It allows the modeling of time-critical aspect in the field of Web Services. It allows incremental composition of services, as well as consistency checking after each modification. It introduces a new type of places named composite places. A composite place is an abstract place represented by a sub-network, allowing a degree of independence between the parts of the H-Service-Net. Indeed, a composite or single place in H-Service-Net depends only on the subnet to which it belongs. In other words, the modification of a component can affect its subnet or the subnets of the same hierarchy. This representation allows for incremental modeling of the H-Service-Net. This will allow for easy correction of errors, an exact location of conflicts between the subnet elements and support rapid changes. Thus, the H-Service-Net model is well suited for modeling the synchronization constraints in a temporal scenario. As a result, it was chosen to model the composition of web services. We present in what follows the different elements of the H-Service-Net model:

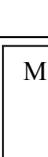
4.1.1 Places in H-Service-Net

Table 1: Places in H-Service-Net.

Place	Modeling	Description
Ordinary Place		It models a basic element (Web service) and its execution time.
Virtual Place		It models a temporal constraint.
Silent Place		It models a place without any specific task, which is used to handle exceptions.
Parallel Place		It models a set of elements of the same group that operate simultaneously, which is used to model concurrent Web service.
Sequential Place		It models a sequence of elements of the same group that run in sequence. It is used to model sequential Web services.
Root Place		It represents the root of the global Petri net and behaves like a sequential composite place.
Loop Place		It is an element that runs in a loop and is used to model a recursive Web service.

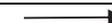
4.1.2 Transitions in H-Service-Net

Table 2: Transitions in H-Service-Net.

Transition	Modeling	Description
Simple transition		It is fired when all its input places are active and have available tokens.
Master transition		It is fired as soon as the place associated with the Master arc is active and has an available token.
First transition		It is fired when one of its input places is active and has a free token.

4.1.3 Tokens and Arcs in H-Service-Net

Table 3: Tokens and arcs in H-Service-Net.

Arc \ token	Modeling	Description
State Token		It defines the state of the Web service associated with the place
Exception Token		It is used to handle exceptions
Simple arc		Control the firing of a simple transition
Master arc		Control the firing of a Master transition.

4.2 Case Study

In this section, we illustrate an application of our system with a real example.

Real Scenario: An orchestration Service of building a house

The time constraints imposed by house building services are very important because in the case of exceeding the expiration time can cause large financial loss. Therefore, verification of temporal consistency is important to overcome this loss.

A composition of service must be provided to satisfy the customer and links must be established with other entities such as: real estate agency, a notary, the mayor, the contractor, a company of selling building materials and finally a bank to perform all financial transactions, for that seven Web service should be invoked:

- Ws1: Estate agency: is an intermediary in transactions concerning real estate.
- Ws2: Ws8: The bank allows performing financial transactions and online payment.
- Ws3: human resource: the notary establishes the act of buying the plot of land between the client and the seller of the plot through the real estate agency.
- WS4: human resource the architect establishes the house plans, according to the customer's requirements.
- WS5: The City Hall: Establishes the permit to build the house.
- WS6: human resource: The contractor: who manages the construction of the house.
- WS7: The sales service of building materials.

For the modeling of these Web Services in a single H-Service-Net, we add the following composite places:

- The online payment service Ws2 and the notary service Ws3 can run in parallel, we model them in H-Service-Net by a parallel composite place P1.
- The architect service Ws4 and the City Hall service

can run in sequence, we model them with a sequential composite place S1.

- The contractor service Ws6 and The building materials sales service ws7 can run in the loop, we model them with a loop composite place L1.
- The estate agency service Ws1 and the Composite services P1, S1, L1 and the online payment service Ws8 run in sequence, we have modeled this set by the root place R. Remember that by definition the elements of the root run in sequence.

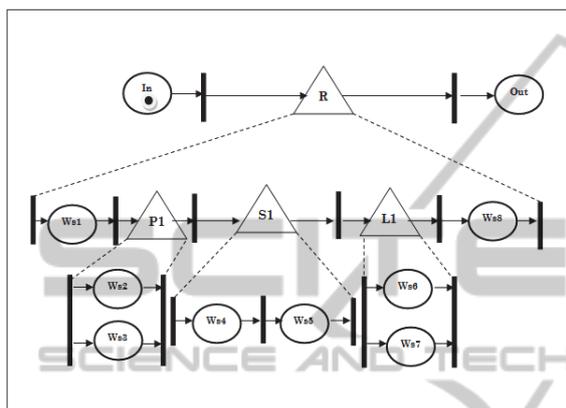


Figure 1: H-Service-Net for the example.

4.3 Temporal Constraint in H-Service-Net Model

In order to complement our system we propose a modular approach to model composition and consider all types of temporal constraints using Extend time unit system (XTUS) (Bouزيد, 2005) to represent temporal interval and the algebra of Allen’s intervals (Allen, 1983) in order to define temporal relations between Web services and comparison operators ($<$, \leq , $>$, \geq , $=$, \neq) for comparing temporal interval to a specific time unit in a time Petri net [(BERTHOMIEU, 1991)] model.

4.3.1 Allen’s Interval Algebra

James F. Allen (Allen, 1983) has defined a set of 13 basic temporal relations between two intervals $\{b, m, o, s, d, f, eq, a, mi, oi, si, di, fi\}$ which describe the relative positions between two times intervals, These relations cover all possible cases of temporal relations.

4.3.2 XTUS (Extend Time Unit System)

XTUS (Bouزيد, 2005) is an algebraic tool for constructing temporal specifications in a simple and powerful way, as XTUS refers to absolute time,

basic time units are defined by providing a full date specification including the following temporal attributes: year, month, day, hour, minute and second.

Although the system is extensible and other BTU (Basic time Unit) can be added, each absolute time can be specified in a unique way as a sequence of integers i with the following form: $i = [year, month, day, hour, minute, second]$. (Bouزيد, 2005)

4.3.3 Schematic Representation of Time Constraints in H-Service-Net

Different measures of time exist to describe the temporal constraints in our model seven time units are adopted so the diagram is defined as follows: [Year, Month, Day, Weekday, Hour, Minute, Second] that allows to express temporal constraints which must be confirmed with calendar dates.

4.3.4 Representation Model of Temporal Constraints

CWS: refers to a composite Web service that is represented as H-Service-Net Model.

$CWS = \{ID, WS, M, R\}$

ID: identifier of the composite Web service.

$WS = \{ws_1, ws_2, \dots, ws_i, \dots, ws_n\}$: Non-empty set of Web service, ws_i : is a Web service component which belongs to the composite Web service CWS.

$M^{?} = \{? m / m \in M^{?}\} \cup \{! m / m \in M^{!}\}$. $M^{?}$: Non-empty set of incoming message, $M^{!}$: Non-empty set of outgoing message.

R: Finite set of temporal constraint, $R = R_1 \cup R_2$.

R_1 : A relation that allows assigning absolute time constraints.

$R_1 = \{XTUS, \varphi_K C_1 \wedge \varphi_K C_2, ws_i \vee M^{?}\}$.

R_2 : A relation that allows assigning relative time constraints between two Web services.

$R_2 = \{ws_i \vee M^{?}, XTUS, RA_K(\varphi_K C_1 \wedge \varphi_K C_2), ws_j \vee M^{?}\}$
 XTUS: Time interval expressed in the XTUS system with the follows schema [Year, Month, Day, Day of Week, Hour, Minute, Second].

K: the number of the temporal attribute.

$K \in \{0, 1, 2, 3, 4, 5\}$.

$k=0$ (BTU="Year", $k=1$ (BTU="Month", $k=2$ (BTU="Day", $k=3$ (BTU="Day of Week", $k=4$ (BTU="Hour", $k=5$ (BTU="Minute", $k=6$ (BTU="Second".

φ_K : A set of comparison operator, $\varphi_K \in \{<_K, \leq_K, >_K, \geq_K, =_K, \neq_K\}$.

C_1, C_2 : Constants that express the values of BTU.

RA_K : Allen’s Relation, $RA \in \{b_k, m_k, o_k, s_k, d_k, f_k,$

$eq_k, a_k, mi_k, oi_k, si_k, di_k, fi_k\}$

The following table represents illustrative examples that express relative temporal constraints using our system for each Allen's relation:

Table 4: Relative Time constraints for each Allen's relation.

Allen Relation	Example
ws_i b ws_j	ws_j can be run between 1 and 5 days after the execution of ws_i $\{ws_i, -, b_2 (> 1 \wedge < 5), ws_j\}$
ws_i m ws_j	ws_j can be run Just after ws_i the first trimester of each year $\{ws_i, [-, [1-3]], m, ws_j\}$
ws_i o ws_j	ws_j can be run at least 20 minutes before the beginning of ws_i $\{ws_i, -, o_s (\geq 20), ws_j\}$
ws_i s ws_j	ws_j can be run at the beginning of ws_i every work day from 8 to 16 hours $\{ws_i, [-, -, -, [2-6], [8-16]], s, ws_j\}$
ws_i d ws_j	ws_j can be run at least 10 minutes before the start of ws_i and more than 20 minutes after the end of ws_i $\{ws_i, -, d_s (\geq 10, \leq 20), ws_j\}$
ws_i f ws_j	ws_i must end its execution by the end of ws_j the last day of each year $\{ws_i, [-, 12, 31], f, ws_j\}$
ws_i eq ws_j	ws_i must begin and end with ws_j each weekend. $\{ws_i, [-, -, -, [0,1]], eq, ws_j\}$

Our model allows expressing all relations of Allen interval algebra. In order to model the temporal units of Web services with time constraints, solutions in section 2 (Diaz, 2006), (Kazhamiakin, 2004), (CHEN, 2011) model time with one basic time unit such as second or an hour. However, all basic time units can be expressed in our solution.

Our system is applicable in every type of service and especially in service aimed at temporal constraints as critical

4.3.5 Temporal Constraint in H-Service-Net

The following temporal constraints are specified

TC1: if the client sends a valid credit card number the notary must send the contract of purchasing the land within a period of 8h.

Temporal constraints can be expressed either by

absolute constraints affected to a single service or by a relative constraint between two messages exchange.

$\{?CCN, -, b_4 (\leq 8), ?Buying - Act\}$.

TC2: in winter, a discount of the price is given to building materials service.

$\{[-, [11-1]], -, ?devis - material\}$.

TC3: service of selling and building materials has its specific delivery time constraints in collaboration with the contractor:

- If the delivery request is received from 8h to 12h material is delivered after at most 1 hour.
- If the delivery request is received from 13h to 16h material is delivered after at least 3 hours.
- If the delivery request is received the weekend the material is delivered over 24 hours.

$\{!ordre - of - delivery, [-, -, -, [0-1]], b_4 (\leq 24), ?delivery\} \cup$
 $\{!ordre - of - delivery, [-, -, -, [8-11]], b_4 (\leq 1), ?delivery\} \cup$
 $\{!ordre - of - delivery, [-, -, -, [12-16]], b_4 (\geq 3), ?delivery\}$

TC4: the client imposes that the construction of the house should not exceed the realization delay of six months.

$\{?accept, -, b_1 (\leq 6), !end - work\}$.

4.3.6 Checking Temporal Consistency at Design Time

This section presents the check-relative-constraint-at-design-time algorithm for verifying relative constraint in composite Web services. It affects the new local constraint to the composite place that contains relative constraint and then checks if the relative constraint between two web services can be satisfied compared to the local constraint of each Web service. For the successful case, the outcome is "total or partial temporal consistency", and for the unsuccessful case the outcome is "temporal inconsistency"

Check-relative-constraint-at-design-time algorithm

ALGORITHM 1. Check-relative-constraint-at-design-time algorithm

Input:

$LC_i = [\min_i, \max_i]_{ki}$: Local constraint affected to ws_i with $BTU=ki$;

$LC_j = [\min_j, \max_j]_{kj}$: Local constraint affected to ws_j with $BTU=kj$;

$RC_{i,j} = \{ws_i, -, R_{k(i,j)} (>1 \wedge < m), ws_j\}$: Relative constraint affected to composite place $TC_{i,j}$ between ws_i and ws_j with $BTU=k(i-j)$;

Return (PTC: partial temporal consistency, TTC: total temporal consistency, TI: temporal inconsistency)

Int Max-k = $\text{Max}(ki, kj, k(i-j))$;

```

s ∈ {i, j, i-j};
Vt convert (V, i, j) {
    T array [6]; T [0] =12; T [1] =30; T [2] =24; T [3] =1;
    T [4] =60; T [5] =60;
    Int t =1;
    For k=i to (j-1)
        {t=t*T [k];}
    Vt=V*t;
    Return Vt; }

```

```

If (ks<Max-k) then
    LCs=[convert(mins,ks,Max-k),convert(maxs,ks,Max-k)];
End if
Switch (R) {
    Case "b": LC [TCi-j] = [mini+l+minj, maxi+m+maxj];
    break
    Case "m": LC [TCi-j] = [mini+minj, maxi+maxj];
    break
    Case "o": LC [TCi-j] = [minj+l, maxj+m];
    Verify-o-constraint (); break
    Case "s, f": LC [TCi-j] = [minj, maxj];
    Verify-s-f-constraint (); break.
    Case "d": LC [TCi-j] = [minj, maxj];
    Verify-d-constraint (); break
    Case "eq": LC [TCi-j] = [max0 (mini, mini), max (maxi,
    maxj)]; Verify-eq-constraint (); break
    }
Verify-o-constraint () {
If (l+minj>mini) & (m+maxj>maxi) then return ("TTC");
Else if (l+minj>mini) then return ("PTC");
    Else if (m+maxj>maxi) then return ("OTC")
    Else return ("TI")
    End if
    End if
End if
Verify-s-f-constraint () {
If (minj-mini>0) & (maxj-maxi>0) then return ("TTC")
Else if (minj-mini>0) then return ("PTC")
    Else if (maxj-maxi>0) then return ("PTC")
    Else return ("TI")
    End if
End if
End if
Verify-d-constraint () {
If (minj>l+mini) & (maxj>maxi+m) then return ("TTC")
Else if ((minj>l+mini) then return ("PTC")
    Else if (maxj>maxi+m) then return ("PTC")
    Else return ("TI")
    End if
End if
Verify-eq-constraint () {
If (mini=minj) & (maxi=maxj) then return ("TTC")
Else if [mini, maxi] ∩ [minj, maxj] ≠ ∅ then return ("PTC")
    Else return ("TI")
    End if
End if
}

```

Example:
Ws₄ and Ws₅ are executed in parallel composite place P₁ with local temporal constraints LC₄=[min₄,max₄], LC₅=[min₅, max₅] and a relative temporal constraint

$$RC_{4-5} = \{Ws_4, R (>l \wedge <m), Ws_5\}.$$

The following table presents some example of executing Check-relative-constraint-at-design-time algorithm

min ₄	max ₄	min ₅	max ₅	R	l	m	LC(P ₁)	result
30	50	40	45	o	10	30	[50,75]	TTC
30	50	5	10	o	10	30	[15,40]	TI
5	20	10	15	eq	10	30	[10,20]	PTC
5	10	20	30	eq	10	30	[20,30]	TI

The following algorithm checks if the client global constraint assigned to a composite place can be satisfied compared to the local constraint of each Web service. For the successful case, the outcome is “total or partial temporal consistency”, and for the unsuccessful case the outcome is “temporal inconsistency”

ALGORITHM 2. Check-global-constraint-at-design-time algorithm

GC= {c, \forall_k c, CompositPlace} **GC**: global constraint affected to a composite place CompositPlace
LC: a set of local time constraint affected to a simple place (atomic Web service) or composite place (composite Web service)
Each simple Place has a local temporal constraint affected by the service provider as a range LC= [min_L, max_L]_k
Return (PTC: partial temporal consistency, TTC: total temporal consistency, TI: temporal inconsistency)

k: the number of the time attribute k ∈ {0,1,2,3,4,5}.
k=0 → BTU="Year", k=1 → BTU="Month",
k=2 → BTU="Day", k=3 → BTU="Day of Week",
k=4 → BTU="Hour", k=5 → BTU="Minute",
k=6 → BTU="Second"

```

maxk max_temporal_attributes () {
maxk=0;
For each simple_place do
    If maxk < simple_place.k then maxk = k;
    End if
And for
Return maxk;
}
Transform_to_interval ( $\forall_k$  c) {.....(5)
If  $\forall_k = <_k$  c then return [0, c-1]
If  $\forall_k = \leq_k$  c then return [0, c]
If  $\forall_k = >_k$  c then return [c+1, +∞]
If  $\forall_k = \geq_k$  c then return [c, +∞]
If  $\forall_k = >_k$  c1 ∧  $\forall_k = <_k$  c2 then return [c1+1, c2-1]
If  $\forall_k = >_k$  c1 ∧  $\forall_k = \leq_k$  c2 then return [c1+1, c2]
If  $\forall_k = \geq_k$  c1 ∧  $\forall_k = <_k$  c2 then return [c1, c2-1]
If  $\forall_k = \geq_k$  c1 ∧  $\forall_k = \leq_k$  c2 then return [c1, c2]
}
For each place do
    Maxk = max_temporal_attribut ().....(1)
    For each place.child do
        LC.child=[convert(minL,k,maxk),
        convert(maxL,k,maxk)].....(2)
    End for
If (place = S) or (place=R)
then LC.place= Somme (LC.child) ....(3)

```

```

And if
If (place = P) then
  LC.place=[Max(minL.child),Max(maxL.child)].....(4)
And if
End for

[ $min_L, max_L$ ] = LC(CompositPlace);
[ $min_g, max_g$ ] = Transform_to_interval (GC);
If ( $[min_g, max_g] \subseteq [min_L, max_L]$ ) then return ("TTC")
Else if ( $[min_L, max_L] \subset [min_g, max_g]$ ) then return ("PTC")
  Else if ( $max_L > min_g$ ) then return ("PTC")
    Else return ("TI");
  End if
End if
End if

If ( $min_L \geq max_g$ ) then return ("TI")
Else return ("PTI");
End if

```

(1) max_temporal_attributes () is a method that returns the largest temporal attribute affected to a simple place (atomic Web service) in the HSN network, i.e. The maximum value of the k variable.
 (2) Temporal attributes conversion of simple place to the value max_k in order to have a common time unit (BTU: Basic Time Unit).
 (3) And (4) calculation methods of local temporal constraints in a composites place.
 (5) The method Transform_to_interval () converts temporal constraint to a temporal interval.

Example:

Ws₄ and Ws₅ are executed in parallel composite place P₁ with local temporal constraints LC₄= [min₄, max₄], LC₅=[min₅, max₅] and a global temporal constraint GC = { -, (>C₁ ∧ <C₂), P₁}.

The following table presents some example of executing Check-global-constraint-at-design-time algorithm

min ₄	max ₄	min ₅	max ₅	C1	C2	LC(P ₁)	result
30	50	40	42	40	45	[40,42]	TTC
30	50	40	45	40	45	[40,50]	PTC
30	50	40	45	10	20	[40,50]	TI

4.3.7 Checking Temporal Consistency at Runtime

This section presents the check-relative-local-constraint-at-run-time algorithm for verifying relative and local constraint at run time in Web service composition

ALGORITHM 3. Check-relative-local-constraint at run time algorithm

LC_i= [min_i, max_i] _{ki}: Local constraint affected to ws_i with BTU=ki;
 LC_j=[min_j,max_j]_{kj} : Local constraint affected to ws_j with BTU=kj;
 RC_{i,j}= {ws_i, -, R_{k (i,j)} (>l ∧ <m), ws_j}: Relative constraint

affected to composite place TC_{i,j} between ws_i and ws_j with BTU=k (i-j);
 Bi, Ei: Begin and End execution of ws_i;
 Bj, Ej: Begin and End execution of ws_j;
Return (LTC: Local temporal consistency, LTI: Local temporal inconsistency, RTC: Relatif temporal consistency, RTI: Relatif temporal inconsistency)

```

If ((Ei-Bi) ∈ [mini, maxi]) then return ("LTC ")
Else return ("LTI");
End if
If (Ej - Bj) ∈ [minj, maxj] ) then return ("LTC")
Else return ("LTI");
End if

```

```

Switch (R)
  Case "b":
    if (Bj ∈ [Bi+mini+1, Bi+maxi+m]) then return ("RTC")
    Else return ("RTI");
    End if
  break
  Case "m":
    if (Bi=Ej) then return ("RTC")
    Else return ("RTI");
    End if
  break
  Case "o":
    if ((Bj ∈ [Bi+1, Bi+m]) & (Ei<Ej) & (Ei ∈ [Bi+1+minj, Bi+m+maxj])) then return ("RTC")
    else return ("RTI");
    End if
  break
  Case "s":
    if ((Bi=Bj) & (Ei < Ej)) then return ("RTC")
    Else return ("RTI");
    End if
  break
  Case "d":
    if ((Bi ∈ [Bj+1, Bj+m]) & (Ei < Ej)) then return ("RTC")
    Else return ("RTI");
    End if
  break
  Case "p":
    if ((Bi<Bj) & (Ei =Ej)) then return ("RTC ")
    Else return ("RTI");
    End if
  break
  Case "eq":
    if ((Bi =Bj) & (Ei=Ej)) then return ("RTC")
    Else return ("RTI");
    End if
  break
End switch

```

Example:

Ws₄ and Ws₅ are executed in parallel composite place P₁ with local temporal constraints LC₄=[min₄,max₄], LC₅=[min₅, max₅] and a relative temporal constraint RC_{4,5}= {Ws₄, R (>l ∧ <m), Ws₅}.

The following table presents some example of

executing Check-relative-local-constraint at run time algorithm

min ₄	max ₄	min ₅	max ₅	R	l	m	B ₄	E ₄	B ₅	E ₅	result
30	40	40	45	o	10	30	35	70	50	95	RTC, LTC
30	40	40	45	o	10	30	35	70	50	65	RTI, LTI

5 CONCLUSIONS

Verification of temporal constraints in Web service composition is an important way to ensure the exactitude and the reliability of composition. Our contribution proposes a modular approach for the modeling composition with time constraint while using Extend time unit system (XTUS) to represent the temporal periodicity of the services, Allen's interval algebra and comparison operators ($<$, \leq , $>$, \geq , $=$, \neq) to represent all types of temporal constraint in a time Petri net model. H-Service-Net model allows an incremental composition of services, as well as checking of temporal consistency at design time, during the execution and after each handling of exception in a simplified manner and allows reducing the quantity of stored data. Finally, we present an algorithm of checking temporal constraint at design and run time.

REFERENCES

- Berthomieu, B., Diaz, M., 1991. Modeling and verification of time dependent systems using time Petri nets. *Software Engineering, IEEE Transactions on (Volume:17, Issue: 3) Page (s): 259 – 273.*
- Diaz, G., Pardo, J., Cambronero, M., Valero, V., Cuartero, F., 2006. Verification of Web Services with Timed Automata. *Notes in Theoretical Computer Science*, vol. 157(2), pp. 19-34.
- Dai, G., Bai, X., 2007. A Framework for Time Consistency Verification for Web Processes Based on Annotated OWL-S. *The Sixth International Conference on Grid and Cooperative Computing(GCC 2007).*
- Allen, J., 1983. Maintaining Knowledge about Temporal Intervals. *Communications of ACM*, Vol 26, no 11, 1983.
- Bouzid, M., Ligeza, A., 2005. hierarchical granular terms and their applications. *In Proc. of the International Conference on Tools with Artificial Intelligence. IEEE, pp. 249–253.*
- Lallali, M., Zaidi, F., Cavalli, A., 2007. Timed Modeling of Web Services Composition for Automatic Testing. *In Proc. Third International IEEE Conference on Signal-Image Technologies and Internet-Based System SITIS '07, pages 417–426.*
- Lallali, M., Zaidi, F., Cavalli, A., 2008. Automatic Timed Test Case Generation for Web Services Composition. *In Proc. IEEE Sixth European Conference on Web Services ECOWS '08, pages 53–62.*
- Rouached, M., Perrin, O., Godart, C., 2006. Towards Formal Verification of Web Service Composition. *in ed. S. Dustdar, J. L. Fiadeiro, and A. Sheth, Lecture Notes in Computer Science 4102: 257–273.*
- Peltz, C. 2003. Web services orchestration and choreography. *Published in: Computer (Volume:36, Issue: 10) Page(s): 46 – 52.*
- Hamadi, R., Benatallah, B., 2003. A Petri Net-based Model for Web Service Composition. *Proceeding ADC '03 Proceedings of the 14th Australasian database conference Volume 17, 2003, Pages 191-200, Australia.*
- Hamadi, R., Benatallah, B., Medjahed, B., 2008. Self-adapting recovery nets for policy-driven exception handling in business processes. *Journal Distributed and Parallel Databases Volume 23 Issue 1, Pages 1 – 44.*
- Kazhamiakin, R., Pandya, P., Pistore, M., 2004. Representation, Verification, and Computation of Timed Properties in Web Service Compositions. *In Proceedings of AAAI Spring Symposium on Semantic Web Services (Stanford University, CA).*
- CHEN, T., GENG, S., 2011. Verification of Time Constraints Consistency on Web Service Composition based on ETPN. *Applied Mechanics and Materials Vols. 58-60 (2011) pp 1094-1099.*