A Colored Petri Nets Model for Diagnosing Semantic Faults of BPEL Services ⋆

Yingmin LI, * **Tarek MELLITI** ** **Philippe DAGUE** *

* *LRI, Univ. Paris-Sud, CNRS, and INRIA Saclay-Île de France F-91893, France (Tel: 33 1 69 72 92 59 93; e-mail: firstname.lastname@lri.fr).*
** *IBISC, Univ. d'Evry Val d'Essonne, CNRS, F-91025, France (e-mail: Tarek.Melliti@ibisc.fr)*

Abstract: The paper contributes to modeling an orchestrated complex Web Service (BPEL) with Colored Petri Nets (CPNs) for diagnosis. In the CPNs model, colored tokens are used to represent the faults in a BPEL process. A uniform fault model is introduced to represent both the faulty input data and external faulty Web services called by the BPEL activities. We use three I/O data dependency relations for each BPEL activity. To represent the fault propagation in colored Petri nets, we define the color propagation functions for each data dependency relation. We give a concrete translation from a BPEL service to a CPNs model. Model-based diagnosis framework is then used. Based on the evolution equation in Petri nets theory, we construct an inequations system as a diagnosis problem and solve it with an algebra algorithm.

Keywords: Model-based diagnosis, BPEL, Web service, Colored Petri Nets

1. INTRODUCTION

Self-healing software is one of the important challenges for Information Society Technologies research. Our paper proposes a centralized diagnosis approach for BPEL (OASIS [2006]) services, whose goal is to design a framework for self-healing Web services by adopting artificial intelligence methodologies to solve the diagnosis problem by supporting online detection and identification of faults.

A Web service (WS) is a set of distributed message oriented interacting components. We can construct complex WS systems by composing basic WSs in two ways: orchestration and choreography (P2P). An orchestrated BPEL service is a central process to organize (basic or complex) WSs to finish complex tasks. A choreographed WS has not a central process while all the involved WSs are aware of their partners but none has the global view of the whole WS application. Distributed WS applications make B2B engineering more convenient but raise more challenges for handling dysfunctions. For example, how to locate the source and reason of faults when they occur somewhere in a distributed WS application? As orchestration is the basic of the WS composition, we focus on single BPEL service diagnosis based on CPN (Diaz [2001]) model which can be easily extended to a distributed environment.

During the interaction of distributed WS components, subtle faults can come from corrupted data or some functional errors. Due to the message oriented nature of WS applications, faulty data is propagated through the execution trace and is used to elaborate other faulty data and control decisions. In this way the subtle faults become large ones. A typical example is a misunderstanding of date format in different languages. 06/03/2009, in English, is June 3, 2009; but in French, is March 6, 2009. If a travel agency WS misinterprets the date format, all the date related reservations will be faulty. Another example is the inconsistent data in data base because of the delay of the WS invocation. Those faults are named as semantic faults.

In this paper, we address the semantic fault by using the model based diagnosis approach, and more specially, the discrete event model based approach (Benveniste et al. [2003], Ardissono et al. [2005], Yan and Dague [2007], Ye and Dague [2008], Li et al. [2007], Chatain and Jard [2005], Zhang et al. [2008], Boukadi et al. [2006]). Among the usual discrete event model, we use colored Petri nets to define the diagnoser. Many works use the Petri nets to do diagnosis (Benveniste et al. [2003], S.Genc and S.Lafortune [2005],Yan and Dague [2007], Ye and Dague [2008], Li et al. [2007]), some of them use high level Petri nets (Chatain and Jard [2005]).

The main originality of this work is a natural use of the colored Petri nets; the color domain is used to model data (states) status and transitions are used to define transition status. The model presented here can be generalized to a very large software domain besides Web services. Another originality is the diagnosis methods: unlike most of other works based on Petri nets, we don't use an unfolding approach (Benveniste et al. [2003], Yan and Dague [2007], Ye and Dague [2008], S.Genc and S.Lafortune [2005]), we use the incidence matrix and the characteristic vector of the observed trace in order to transform the diagnosis problem to an inequations system, and then we propose an

⋆ This research is supported by EU through the FP6 IST project 516933, WS-DIAMOND (Web Services DIAgnosability, MONitoring and Diagnosis) and national ANR project WEBMOV (Web services Modeling and Verification).

algorithm to solve an inequation and then the inequations system.

The paper is organized as follows: in section 2, we introduce CPN model for the BPEL services and define their firing rules. We define CPN model for typical basic activities and structural operators of BPEL in section 3; in section 4, we define the diagnosis problem and its solution and illustrate it with a concrete example; in section 5, we introduce some related work, compare the different methods, and give some directions for future research.

1.1 Example: flight agent

Before the theoretical part, we describe an orchestrated BPEL example which will be used as an illustration example along this paper.

A BPEL service $flightAgent$ calculates a series of business flight costs. The $flightAgent$ starts with a receive activity C to receive a request string of the series of departure cities and dates, for example, from Paris to London on 01/03/2008, and from London to Madrid on 03/03/2008, from Madrid to Rome on 05/03/2008, and from Rome to Paris on 09/03/2008, all the dates are in French format. $FlightAgent$ iteratively (by using While activity W) invokes an invoke activity S to split the request string to get the information for one flight: the departure city, a departure date, and an arriving city (which is also the departure city of next flight). Whereafter an invoke activity O reserves the flight tickets and cumulatively calculates the flight fees. As to different date formats in English and French, 01/03/2008, 03/03/2008, 05/03/2008, and 09/03/2008 are misinterpreted as January 03, 2008, March 03, 2008, May 03, 2008, and September 03, 2008. In the next, reply activity P returns the total ticket price of the whole trip (in figure 3a). If the client (or other Web service which invokes $flightAgent$) finds the dates on the flight tickets are not correct, or the travel fees are too huge, an exception occurs due to a semantic fault.

2. COLORED PETRI NET

A Petri net is a Colored Petri Net if its tokens can be distinguished by colors. Here we restrict the definition of Colored Petri Net that we use in this paper.

Let E be a set, a multiset on E is an application m from E to $\mathbb{Z}$ (a multiset is denoted as $m = q_0e_0 + ... + q_ne_n$ where $q_i = m(e_i)$). We use $\mathcal{M}(E)$ to define the set of finite multisets from E to $\mathbb{Z}$, and $\mathcal{M}^+(E)$ if we restrict it to $\mathbb{N}$. Sum and subtract operators between two multisets are defined as in Jensen [1997]. For two given value domains D, D', we denote by $[D \rightarrow D']$ the set of possible functions from D to D'.

Definition 1. A Colored Petri Net graph (CPN graph) is a tuple N=$\langle\Sigma,\mathcal{X},F,P,T,cd,Pre,Post\rangle$, where: Σ is a set of colors (see Jensen [1997]). $\mathcal{X}$ is set of variables that range over Σ. F is a set of color functions, $F \subseteq \bigcup_n [\Sigma^n \rightarrow \Sigma]$. P is a set of labeled places, and there are two types of places exists: AP, the activation places which contains the CPN execution control, DP, which contains the data used during the execution of CPN, especially, we denote the constant data places set as CP; Formally, this is represented as follows: $P : AP \cup DP$ and $CP \subseteq DP$, $AP \cap DP = \emptyset$. T is a set of labeled transitions, we denote $Type : T' \rightarrow T''$ with $T', T'' \subset T$ and $T' \cap T''$=$\emptyset$ is a type function of T. $Cd : P \rightarrow 2^{\Sigma}$, is a function that associates to each place a color domain [1] . $Pre, Post$: are forward and backward matrices such that $Pre : P \times T \rightarrow \mathcal{M}^+(\Sigma \cup \mathcal{X})$, are input arc expressions. And $Post : P \times T \rightarrow \mathcal{M}^+(\mathcal{E})$, are output arc expressions.

$\mathcal{E}$ represents a color expression which can be a color constant, a variable, or a color function of F (completely or partially instantiated). Given an expression $e \in \mathcal{E}$, we use $Var(e)$ to denote the set of variables which appear in e, and $Eval(e)$, the evaluation of e in Σ.

We denote $^\bullet t$ and $t^\bullet$ as the input and output places set of transition t, $^\bullet p$ and $p^\bullet$ as the input and output transitions set of place p.

Definition 2. A CPN graph $N = \langle\Sigma,\mathcal{X},F,P,T,cd,Pre,Post\rangle$ is well formed iff: $\forall t \in T, \forall p \in t^\bullet$, we have $Var(Post(p,t)) \subseteq Var(Pre(.,t))$ with $Var(Pre(.,t)) = \bigcup_{p' \in ^\bullet t} var(Pre(p',t))$.

In a well formed CPN graph, we restrict that for each transition, the output arc expressions must be composed by the variables which are in the input arcs expressions.

To each CPN graph, we associate its terms incidence Matrix C ($P \times T \rightarrow \mathcal{M}(\mathcal{E})$) with $C = Post - Pre$.

In the following, we define the behaviors (the dynamics) of a CPN System.

Definition 3. A marking M of a CPN graph is a multiset vector indexed by P, where $\forall p \in P, M(p) \in \mathcal{M}^+(cd(p))$.

Operators + and − on multisets are extended to markings in an obvious way.

Definition 4. A Colored Petri Net system (CPN system) is a pair S=$\langle N, M_0\rangle$ where N is a CPN graph and M_0 is an initial marking.

Definition 5. A transition t is enabled in a CPN system S with present marking M, iff $\exists u$, with $M \geq Pre(.,t)^u$, $Var(Pre(.,t)) \rightarrow \Sigma$, which is a binding of the input arcs variables. [2]

We use $M[t\rangle^u$ to denote that t is enabled in M by the use of u, and we use the classic notation $M[t\rangle$ if u is not important (e.g. when u is unique).

Definition 6. Let M be a marking and t a transition, with $M[t\rangle^u$ for some u. The firing of the transition t changes the marking of CPN from M to $M' = M + C(.,t)^u$. We note the firing as $M[t\rangle^u M'$.

Definition 7. We extend the definition 6 to a sequence of transitions $\delta \in T^*$ as: $M[\delta\rangle M$ if δ is the empty sequence; $M[\omega t\rangle M'$ iff $\exists M''$ such that $M[\omega\rangle M''$ and $M''[t\rangle^u M'$.

3. FROM BPEL TO CPN MODEL

There exist already many works dedicate to translate BPEL services into CPN model for verifying (Tan et al.

[1] In this definition, a transition has no color domain. This restriction will be explained in section 3.2.

[2] u must respect the color domain of the places, i.e., $\forall p \in ^\bullet t$, $x \in var(Pre(p,t))$, we have $u(x) \in cd(p)$.

[2009],Boukadi et al. [2006]), composing (Zhang et al. [2008]), supervising (Chatain and Jard [2005]), etc.. In this section, we construct our own CPN model by introducing the faulty behaviors into Petri nets model which is suitable not only for diagnosing BPEL services, but also for diagnosing other large software systems.

A BPEL process consists of basic activities and structured operators. The idea of modeling BPEL to CPN is: to map each primitive data to a place, each basic activity to a transition. To each basic activity, input and output activation places $a^{in} \in P$ and $a^{out} \in P$ are associated to identify the execution order. To include the fault model, additional transitions are added to represent the unobservable faulty activities either in basic WSs or in BPEL services. The structured operators are modeled as CPNs which sew the structured sub-processes by combining, disjointing, or generating the local activation places. Once a red token is generated by a faulty transition in a basic activity, the fault is passed along the execution trace through the arc expressions which are represented in *Pre* and *Post* matrices. In the following, we define how to translate the static and dynamic features into CPN models.

3.1 BPEL data Variables and constants

BPEL data variables and constants
To catch maximally the dependency between data (variables, constants, etc.), we decompose the structured data types into their elementary parts, denoted by the *leaves* of their XML tree structure. For a variable X of type m (resp. an Xpath expression), we use x_i to range over the $Leaves(m)$ (resp. $Leaves(X)$) and denote the x_i part of X by a couple (X, x_i). In our mapping, each data variable and constant is represented by a unique place in CPNs.

Color Domain
In our CPN model, three colors are used: red (r) marks a place with faulty data value; black (b), not faulty data value; and unknown color ($*$), unknown correctness of data value.

Data dependency within BPEL v.s. color functions
To specify the effect of each activity on data, we give each activity a data dependency signature in term of three dependency relations (Ardissono et al. [2005]): forward (FW), if the activity just copies the value from the input to the output; source (SRC), if the output data is generated by the activity; and elaboration (EL), if the output data is elaborated from the set of input data. To each of this dependency relation, we associate a color propagation function to represent the data status (faulty, correct, or unknown status) production.

Definition 8. Given the data relations set $D = \{FW, SRC, EL\}$, $\forall d \in D$, the associated color propagation function d^c is defined as: $\forall c, c' \in \Sigma, \forall \mathcal{C} \subseteq \Sigma$,

$$\begin{cases} FW^c \in [\Sigma \rightarrow \Sigma],\ FW^c(c){=}c \\ SRC^c \in [\emptyset \rightarrow \Sigma],\ SRC^c{=}* \\ EL^c \in [2^{\Sigma} \rightarrow \Sigma],\ EL^c(\mathcal{C}){=}c',\ \text{with } c'= \\ \quad \begin{cases} b, \text{ iff } \forall c \in \mathcal{C},\ c{=}b \\ r, \text{ iff } \exists c \in \mathcal{C},\ c{=}r \\ *, \text{ iff } \exists c \in \mathcal{C},\ c{=}* \wedge \nexists c'' \in \mathcal{C},\ c''{=}r \end{cases} \end{cases}$$

In the following sections, we model dynamic features, the basic BPEL activities and structured operators with CPNs.

3.2 Translate basic BPEL activities into CPNs

BPEL service is composed with a series of basic activities. We map each basic activity to its CPN model. Due to space limitation, we restrict our definitions to four main basic activities (*Receive*,*Assign*,*Invoke*, and *Reply*) while the other similar activities can be easily translated in the same way.

The main idea in mapping BPEL basic activities to CPNs is: each primitive data is mapped to a place, each basic activity is mapped to a transition, and *Pre* and *Post* matrices are defined based on semantical data dependency. In order to distinguish the activities execution order and the traces among different branches, to each basic activity, we associate an input activation place a^{in} and an output activation place a^{out}.

As we focus on the semantic fault diagnosis of one BPEL service, the BPEL service code is assumed to be correct. Possible faults can be faulty data received by *Receive* activities, or faulty activities which come from other WS called by *Invoke* activities. So we must introduce fault models for *Receive* and *Invoke* activities to localize the faulty data or external WS. Our approach is to introduce additional transitions to represent the unobservable faulty activities and to define the color functions in *Pre* and *Post* matrices which represent the propagation of faults.

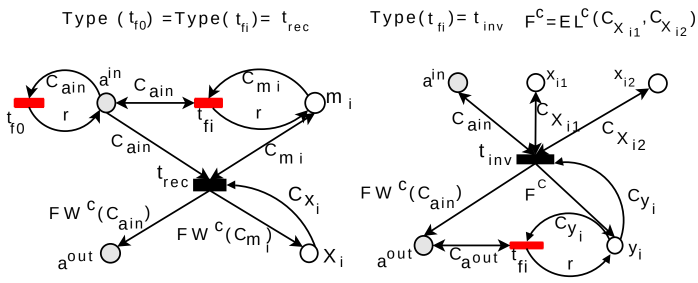

Fig. 1. receive and invoke CPN

Receive(m,X): an activity simply copies the values from a message m to a local variable X. In order to model the receiving of a set of faulty parts from a message value, we add for each part of the message an internal transition (fault) before the firing of the receive transition in figure 1 (left). Note that data places (m, m_i), (x, x_i) are simplified as m_i, x_i.

The CPN model of *Receive* contains two kinds of fault transitions: the activation fault transition t_{f_0}, and the data fault transitions t_{f_i}, we define their types as: $Type(t_{f_0}) {=} Type(t_{f_i}) {=} t_{rec}$. The execution of t_{f_i} is triggered by the consummation of the token in the input activation place. Once t_{f_0} (or t_{f_i}) is executed, we can deduce that there is a faulty control (or data) input. The transmission of the fault (red token) is illustrated on the arc expressions. Each arc expression represents the colored token consumed (on an arc (p, t)) or produced (on an arc (t, p)). To keep the liveness of the CPNs, we add an arc from the output place x_i to the receive transition t_{rec} and its associated color function C_{x_i} is the color of the output data place x_i.

Reply(Y,m): an activity that copies values from a variable Y to a message m for returning the response of the BPEL service to its invoker. So *Reply* can be the ending of BPEL and simply forwards (FW) values. There is no fault model in its CPN and we simply fill *Post* with FW functions.

Assign(X,Y): an activity that reorganizes local variable parts inside a BPEL process without changing the values. So its model is similar to *Reply* activity. Similar operators: *Throw* and *Rethrow*. The *Wait*, *Empty*, and *Exit* activities do not have relation with the variables, so their CPN model only have the input and output activation places.

Invoke(X,Y): an activity that calls another basic or composite Web service. It takes the value of the variable X as input and stores the output in the variable Y. The data dependency can be FW, EL, and/or SRC. As Y can be infected by external faulty WS which is unobservable, we introduce a series of unobservable faulty transitions after the *invoke* transition to model the faults caused by external WS as is illustrated in figure 1 (right).

The CPN model of *Invoke* only contains the data fault transitions t_{f_i}, which are triggered by the consummation of the token in the output activation place. Once t_{f_i} is executed, there should be a fault in its output data place and it can be passed to the other activities along the BPEL process execution trace. Again, we define $Type(t_{f_i}) = t_{inv}$.

3.3 Translate structured BPEL activities into CPNs

In this section, we show how to obtain BPEL process CPN by a modular combination of a set of CPNs. We formally define four main structural operators (*Sequence*, *Switch*, *While*, and *Flow*) while the other similar operators can be easily translated in the same way.

Sequence operator $sequence(S_1, S_2)$
Sequence connects different activities, and the execution order of these activities is the same as their appearance order in the constructor. So we can generate the resulting sequence CPN by simply merging the local intermediate output and input activation places of contractive CPNs (in figure 2(a)).

Conditional operator $Switch(\{(con_i(\overline{X_i},\overline{V_i}),S_i)\}_{i\in I})$
Switch represents an alternative execution of the activities S_i under the conditions $con_i(\overline{X_i},\overline{V_i})$. $\overline{X_i}$ and $\overline{V_i}$ are respectively the variables and constants. For each subprocess S_i, we add a transition con_i to generate its activation place. Each con_i takes the common activation input place of *Switch*, $\overline{X_i}$, and $\overline{V_i}$ as inputs to elaborate an input activation place a_i^{in} for subprocess S_i. A new a^{out} is added to replace all the a^{out_i} of subprocess S_i (in figure 2(c)). Similar operators: *Scope* together with the compensation handlers, event handlers, and fault handlers, *Pick* together with *OnMessage*, *IF*, *Link*.

Iterative operator $while(con(\overline{X},\overline{V}), S_1)$
While iterates the activity S_1 execution until the breaking off of the conditions $con(\overline{X})$. The CPN graph of *While* is similar to *Switch* in which the activation input place of the subprocess S_1 is elaborated by the activation input place of *While*, $\overline{X}$, and $\overline{V}$. But in *While*, the a^{out} of iterative subprocess is also a^{in} of t_{con}. Note that t_{con} represents the transition if condition *con* is true and $t_{\overline{con}}$ represents the transition if condition *con* is false (in figure 2(b)). Similar operators: *RepeatUntil*, *ForEach*.

Parallel operator $flow(\{S_i\}_{i\in I})$
Flow executes the activities S_i in parallel. It terminates when all the activities are finished (fork-join). So we add a^{in}, a^{out}, t^{in}, and t^{out} to compose the subprocesses together in parallel (in figure 2(d)).

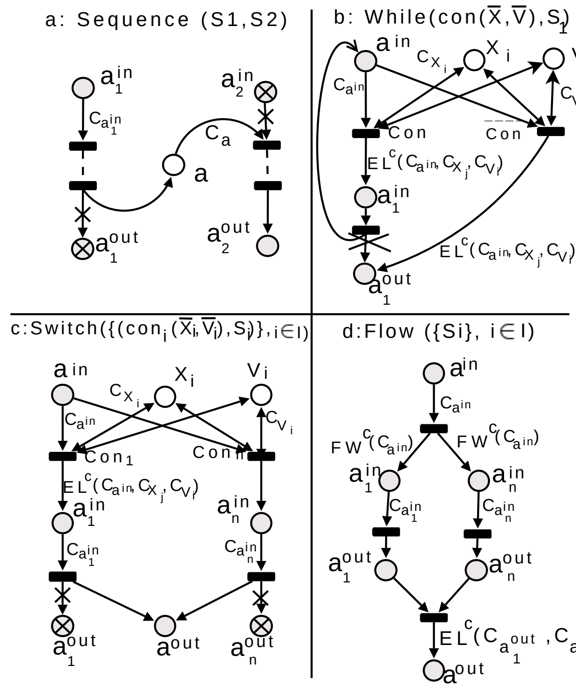

Fig. 2. CPN models of the structural operators

3.4 Some remarks on the BPEL model

Observable vs unobservable transitions
To distinguish the BPEL activities transitions which are observable and the fault transitions which are not, we divide T into observable transitions T_{obs} and fault transitions T_F (T=$T_{obs}\cup T_F$ and $T_{obs}\cap T_F$=$\emptyset$). Remember a type function over faults has been defined, that associates to a fault its observable transition Type: $T_F\rightarrow T_{obs}$.

Initial and symptom markings of BPEL model net
The initial marking is obtained by marking P. CP are marked as unknown as they cannot be changed by any transition; DP are marked as black; a $ap \in AP$ which activate the first execution of CPN is marked as black and the other AP are marked as 0. The final marking is retrieved from the thrown exception. When fault(s) occurs, an exception will be thrown to specify on which activity, there is a faulty part(s), which corresponds to the places in DP. Specially, un matched or uninitiated data (variable) refers the BPEL process may chose fault execution branch. In this case, the input activation place of the activity will be marked as r. All the other places are marked as unknown because there is no information of their marking.

One-boundness of the BPEL model nets
The resulted CPNs are one-bounded (or safe, means one

place can at most contain one token, proof is omitted because of space limitation), in which places represent either data or activation variables.

3.5 Example (cont'd):

Now we can construct the CPN of the BPEL service *flightAgent* as in figure 3. Note that place d_0 represents the request string, d_1 is a null flight schedule variable, and invoke activity t_o will fill it with data during the execution of process *flightAgent*. Place d_5 is the output flight schedule list variable. To keep the visibility of the graph, the color functions which do not concern the data dependency are omitted, for example, color function $C_{a^{in}}$ on the arc (a^{in}, t_c) is omitted.

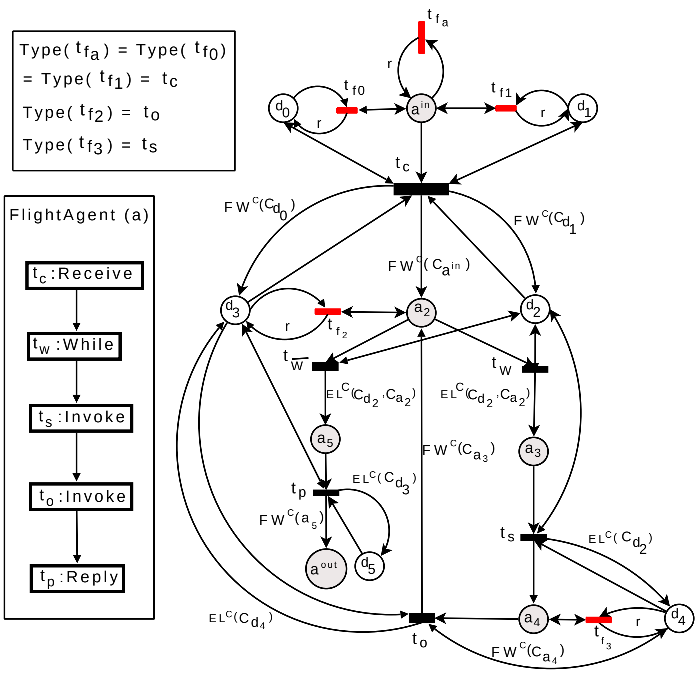

Fig. 3. BPEL (a) and CPN model of *flightAgent*

4. DIAGNOSIS OF BPEL SERVICE USING CPN

4.1 Diagnosis problem

During the execution of a BPEL service instance, we can record the sequence of activities executed within this instance, that we call the trace. This trace belongs to $(T_{obs})^*$. When a fault occurs at some moment of the instance execution, an exception is thrown, what we call in diagnosis literature, a symptom. Exceptions are thrown due to some inconsistency of a part of the services state. The inconsistency can concern either data variables values or activation data (e.g receiving a bad message, or not receiving an expected message). In both cases, a thrown exception can be represented as a marking where the faulty data (or activation) places are marked with a red token and the others can be marked either as black or unknown.

Definition 9. Let M be a marking, M is a symptom (exception) marking iff $\exists p$, $M(p)(r){\neq}0$. We denote the symptom markings by $\hat{M}$.

We can now give the definition of a diagnosis problem as follows:

Definition 10. A diagnosis problem is a tuple $\mathcal{D}$=$< N, \delta_o, \hat{M} >$:

- N is a CPN system that represents the model of a BPEL service;
- δ_o is an observable trace $\delta_o \in (T_{obs})^*$;
- $\hat{M}$ is a symptom marking.

Before giving a definition of a solution to a diagnosis problem, we introduce a covering relation as follows:

Definition 11. A covering relation $\preceq$ between colors of Σ=$\{r, b, *\}$ is a partial ordered relation where any color covers itself and the $*$ color covers all colors (i.e $\preceq$=$\{(r, r)$, (b, b),$(*, *)$,$(r, *)$,$(b, *)\}$). We extend the color covering relation to multisets and markings as follows:

- let $m, m' \in \mathcal{M}^+(\Sigma)$, we have $m \preceq m'$ iff $\sum_{c\in\Sigma} m(c)$=$\sum_{c\in\Sigma} m'(c) \wedge$ $\forall c \neq *,\ m'(c) > 0 \Rightarrow m(c) \geq m'(c)$
- let M, M' be two markings, we have $M \preceq M'$ iff $\forall p \in P$, $M(p) \preceq M'(p)$

We give now a definition of a diagnosis:

Definition 12. Let $\mathcal{D}$=$\langle N$,δ_o,$\hat{M}\rangle$ be a diagnosis problem, a diagnosis $Sol{\subseteq}T_F$ and $Sol{\neq}\emptyset$ such that: M_0+$C\times\overrightarrow{\delta}\preceq\hat{M}$ with $\overrightarrow{\delta}$ is a characteristic vector defined as follows:

- $\forall t \in T_{obs}$, $\overrightarrow{\delta}(t)$=$\overrightarrow{\delta_o}(t)$, where $\overrightarrow{\delta_o}(t)$ is the occurrence number of t in δ_o;
- $\forall t_f \in Sol$, $\overrightarrow{\delta}(t_f)$=1;
- $\forall t_f \in (T_F \setminus Sol)$, $\overrightarrow{\delta}(t_f)$=0.

Note that we restrict the value of a fault transition to 1. This is due to the fact that a fault transition only changes the color of token to red and has no effect on the activation places marking. Even if a fault happens more than once we consider only the occurrence of the fault transition that can explain the symptom (the red token). Thus we restrict the value of the characteristic vector of a fault transition to one or zero (happened and explains the symptom or did not happen).

Definition 13. Let $\mathcal{D}$=$\langle N$,δ_o,$\hat{M}\rangle$ be a diagnosis problem and Sol be a diagnosis, Sol is minimal iff $\forall Sol' \subset Sol$, Sol' is not a diagnosis.

Definition 14. Let $\mathcal{D}$=$\langle N$,δ_o,$\hat{M}\rangle$ be a diagnosis problem, the diagnosis solution $\mathcal{DS} \subseteq 2^F$ is the set of all possible minimal diagnoses.

4.2 Diagnosis of CPN by inequations system solving

Let $\mathcal{D}$=$\langle N$,δ_o,$\hat{M}\rangle$ be a diagnosis problem and let n_i be variables ranging over $\{0, 1\}$, we construct the characteristic vector δ as follows:

- $\forall t \in T_{obs}$, $\overrightarrow{\delta}(t)$=$\overrightarrow{\delta_o}(t)$;
- $\forall t_{f_i} \in T_F \wedge \overrightarrow{\delta_o}(Type(t_{f_i})) \neq 0$, $\overrightarrow{\delta}(t_{f_i})$=$n_i$;
- $\forall t_f \in T_F \wedge \overrightarrow{\delta_o}(Type(t_f))$=0, $\overrightarrow{\delta}(t_f)$=0;

We can then construct an inequations system (one inequation for each place) for the diagnosis problem as follows:

$$Q_{\hat{M}} = \begin{cases} Eq_{p_1} : \hat{M}(p_1) \succeq M_0(p_1) + C(p_1, .)\overrightarrow{\delta} \\ \dots \\ Eq_{p_i} : \hat{M}(p_i) \succeq M_0(p_i) + C(p_i, .)\overrightarrow{\delta} \\ \dots \end{cases}$$

To each place p, we associate an inequation Eq_p where the left part is $l(Eq_p)=\hat{M}(p)$ and the right part is $r(Eq_p)=M_0(p)+C(p,.)\overrightarrow{\delta}$. We divide the set of inequations $Q_{\hat{M}}$ into three subsets:

- $Q^r_{\hat{M}}=\{Eq_p|l(Eq_p)=r\}$
- $Q^b_{\hat{M}}=\{Eq_p|l(Eq_p)=b\}$
- $Q^*_{\hat{M}}=\{Eq_p|l(Eq_p)=* \vee l(Eq_p)=0\}$

The diagnosis algorithm executes backward reasoning recursively (algorithm 2) for each inequation $Eq_p \in Q^r_{\hat{M}}$ within $Q_{\hat{M}}$ and then combines all the diagnosis results (algorithm 3). In the following, we give first the solution of one inequation and then that of an inequations system.

One inequation $Q^r_{\hat{M}}$ solving

The part on the right side of an inequation is a multi set composed by color functions, constants, and the corresponding place variables which may have positive or negative coefficients. Solving the inequation consists in canceling the negative terms in the right part, keeping the positive color functions, and evaluating the positive coefficient n_i red tokens to 1 (algorithm 1). Algorithm 1 looks for the possible minimal diagnosis N^r_p corresponding to one symptom place p in a symptom marking. And at the same time, it looks for the candidate inequations C^r_p which can explain the symptom place but should be solved further. So to completely solve an inequation, we need to recursively solve C^r_p until getting a final diagnosis solution for one symptom place. The idea is to recursively solve each inequation in $Q^r_{\hat{M}}$ by getting the diagnosis solution Sol_p for one symptom place (algorithm 2).

Algorithm 1 Algorithm partially solving a $Q^r_{\hat{M}}$ inequation: $solvAnEqu(Eq_p)$

Input: Eq_p: a $Q^r_{\hat{M}}$ inequation concerns a place p;
Output: $< C^r_p, N^r_p >$ {C^r_p:a set of color functions which generate red tokens; N^r_p: a set of faulty transitions;}
$C^r_p=\emptyset$; $N^r_p=\emptyset$;
ForEach $n_i \times c_i \in r(Eq_p)^+=\sum_{i\in I} n_i \times c_i$ **do**
  if n_i is not a constant and $c_i = r$ **then**
    $N^r_p=N^r_p \cup \{t_{f_i}\}$; {records the faulty transition t_{f_i} in N^r_p}
  else if c_i is a color function concerning place p' **then**
    $C^r_p=C^r_p \cup \{c_{p'}\}$;{records the place $c_{p'}$ if its color c_i is unknown for further solving}
  else if c_i is a color propagation function d^c_i **then**
    $C^r_p=\{C^r_p\} \cup \{c_{p_i} \in Var(c_i)\}$;{records all the input places of c_i for further solving}
  end if
end for
return $< C^r_p, N^r_p >$;

An inequations system $Q_{\hat{M}}$ solving

By solving each inequation in $Q^r_{\hat{M}}$, we get the diagnosis for a inequations system $Q_{\hat{M}}$ (algorithm 3). The union set of all the Sol_p is the diagnosis solution for $Q_{\hat{M}}$ which can contain multiple symptoms (faults).

4.3 Example (cont'): incidence matrix of flight agent

In the example of CPN of flight agent, we can see that flight agent CPN contains 12 places and 11 transitions (5 of them are unobservable, and 6 are observable). Table 2 is the forward matrix, table 1 is the backward matrix, and table 3 is the incidence matrix of flight agent got by $C=C^+ - C^-$. As to space limitation, in the incidence matrices, we use the name of places to represent the colors of the places, for example, a^{in} represents $C_{a^{in}}$. Transitions $t_{f_a}, t_{f_0}, t_{f_1}, t_{f_2}$, and t_{f_3} are unobservable activities (in gray columns). Especially t_{f_a}, t_{f_0}, and t_{f_1} generate the input

[3] $\overset{\cup}{\times}$ is an operator that applies the union operator on couples resulting from the Cartesian product.

Algorithm 2 Diagnosis solution algorithm for completely solving a $Q^r_{\hat{M}}$ inequation: $CSD(Q_{\hat{M}}, Eq_p)$

Input: $Q_{\hat{M}}=Q^r_{\hat{M}} \cup Q^b_{\hat{M}} \cup Q^*_{\hat{M}}$: the inequations system ;
$Eq_p \in Q^r_{\hat{M}}$: an inequation to solve;
Output: Sol_p: a diagnosis solution concerning a symptom place p;
$Sol_p=\emptyset$;
$\langle C^r_p, N^r_p\rangle=solvAnEqu(Eq_p)$;{get the first back reasoning result, C^r_p need to be resolve further}
$Sol_p=Sol_p \cup N^r_p$;{record the current diagnosis}
if $C^r_p \neq \emptyset$ **then**
  ForEach $c_{p'} \in C^r_p$ **do**
    if $\exists Eq_{p'} \in Q^*_{\hat{M}}$ **then**
      if $l(Eq_{p'})=*$ **then**
        $Sol_p = Sol_p \cup CSD(Q^r_{\hat{M}} \cup \{r \succeq r(Eq_{p'})\} \cup (Q^b_{\hat{M}} \cup Q^*_{\hat{M}}) \setminus \{Eq_p, Eq_{p'}\}, r \succeq r(Eq_{p'}))$; {evaluates the $l(Eq_{p'})$ as r, reconstructs the inequations system and recursively back reasoning until solved all the related places}
      else if $l(Eq_{p'})=0$ **then**
        $Sol_p=Sol_p \cup CSD(Q^r_{\hat{M}} \cup \{r \succeq r(Eq_{p'})+c_{p'}\} \cup (Q^b_{\hat{M}} \cup Q^*_{\hat{M}}) \setminus \{Eq_p, Eq_{p'}\}, r \succeq r(Eq_{p'}) + c_{p'})$;{evaluates the $l(Eq_{p'})$ as r and add a red token on the right side of the inequation to balance $Eq_{p'}$, reconstructs the inequations system, and recursively back reasoning until solved all the related places}
      end if
    end if
  end for
end if
return Sol_p;

Algorithm 3 Diagnosis solution algorithm for $Q_{\hat{M}}$

Input: $Q_{\hat{M}}=Q^r_{\hat{M}} \cup Q^b_{\hat{M}} \cup Q^*_{\hat{M}}$: the inequations system ;
$Sol_p=\emptyset$: a diagnosis solution concerning a symptom place p;
Output: D: a diagnosis solution of $Q_{\hat{M}}$;
$D=\emptyset$;
ForEach $Eq_p \in Q^r_{\hat{M}}$ **do**
  $Sol_p=CSD(Q_{\hat{M}}, Eq_p)$; {resolve each inequation in $Q^r_{\hat{M}}$ by back reasoning}
  $D=D \overset{\cup}{\times} Sol_p$;[3]
end for
return D;

Table 1. C^-: backward matrix of flight agent

C^-	t_{f_a}	t_{f_0}	t_{f_1}	t_C	t_W	$t_{\overline{W}}$	t_S	t_{f_3}	t_O	t_{f_2}	t_P
a^{in}	a^{in}	a^{in}	a^{in}	a^{in}							
a_2					a_2	a_2					
a_3							a_3			a_3	
a_4								a_4	a_4		
a_5											a_5
a^{out}											
d_0		d_0		d_0							
d_1			d_1	d_1							
d_2				d_2	d_2	d_2	d_2			d_2	
d_3				d_3					d_3		d_3
d_4							d_4	d_4	d_4		
d_5											d_5

Table 2. C^{+}: forward matrix of flight agent

C^{+}	t_{f_a}	t_{f_0}	t_{f_1}	t_C	t_W	$t_{\overline{W}}$	t_S	t_{f_3}	t_O	t_{f_2}	t_P
a^{in}	r	a^{in}	a^{in}								
a_2				$FW^c(a^{in})$					$FW^c(a_4)$		
a_3					$EL^c(a_2, d_2)$					a_3	
a_4							$FW^c(a_3)$	a_4			
a_5						$EL^c(a_2, d_2)$					
a^{out}											$FW^c(a_5)$
d_0		r		d_0							
d_1			r	d_0							
d_2				$FW^c(d_0)$	d_2	d_2	d_2			r	
d_3				$FW^c(d_1)$					$EL^c(d_4)$		d_3
d_4							$EL^c(d_2)$	r	d_4		
d_5											$EL^c(d_3)$

Table 3. $C = C^{+} - C^{-}$: incidence matrix of flight agent

$C^{+} - C^{-}$	t_{f_a}	t_{f_0}	t_{f_1}	t_C	t_W	$t_{\overline{W}}$	t_S	t_{f_3}	t_O	t_{f_2}	t_P
a^{in}	$r - a^{in}$			$-a^{in}$							
a_2				$FW^c(a^{in})$	$-a_2$	$-a_2$			$FW^c(a_4)$		
a_3					$EL^c(a_2\ ,\ d_2)$		$-a_3$				
a_4							$FW^c(a_3)$		$-a_4$		
a_5						$EL^c(a_2\ ,\ d_2)$					$-a_5$
a^{out}											$FW^c(a_5)$
d_0		$r - d_0$									
d_1			$r - d_1$								
d_2				$FW^c(d_0)$-d_2						$r - d_2$	
d_3				$FW^c(d_1)$-d_3					$EL^c(d_4)$-d_3		$-d_3$
d_4							$EL^c(d_2)$-d_4	r- d_4			
d_5											$EL^c(d_3)$-d_5

fault data of flight agent, t_{f_3} represents the external fault in the WS which is invoked by t_S, and t_{f_2} represents the external fault in the WS which is invoked by t_O.

4.4 Example (cont'): diagnosis solution of flight agent

In our diagnosis scenario, each BPEL process is associated with a monitoring platform, which dedicates to record the status of the activities and variables of each execution instance, and a diagnosis WS, which contains the initialed (all DP are marked as black and AP are marked as 0 excepts the first input activation) CPN model of the BPEL and acknowledge the diagnosis WS. The diagnosis WS can be triggered by the BPEL executer (BPEL execution engine) or invoker (WS, application, etc). Once a symptom is thrown by the executer or invoker, the (activation or data) places which correspond to the symptom is marked as r while the other data places are marked as $*$, and activation places are marked as 0. Now suppose we get a series of observed activities σ_0: C, W, S, O, W, S, O, $\overline{W}$, and P, which means the while iteration is processed twice. Then we construct a characteristic vector $\overrightarrow{\delta}^T$: ($t_{f_a}$ t_{f_0} t_{f_1} t_C t_W $t_{\overline{W}}$ t_S t_{f_3} t_O t_{f_2} t_P)=(n_0 n_1 n_2 1 2 1 2 n_4 2 n_3 1). Given an initial marking M_0 =(a^{in} a_2 a_3 a_4 a_5 a^{out} d_0 d_1 d_2 d_3 d_4 d_5)=(b 0 0 0 0 0 b b b b b b), we suppose that, in two diagnosis scenarios, we got two symptom markings M_{n_1} = (a^{in} a_2 a_3 a_4 a_5 a^{out} d_0 d_1 d_2 d_3 d_4 d_5)=(0 0 0 0 0 $*$ $*$ $*$ $*$ $*$ $*$ r), and M_{n_2} = (a^{in} a_2 a_3 a_4 a_5 a^{out} d_0 d_1 d_2 d_3 d_4 d_5)=(0 0 0 0 0 r $*$ $*$ $*$ $*$ $*$ r). For symptom marking M_{n_1}, we can construct an inequations system as in (1).

Note that for final marking M_{n_2}, we can construct a similar inequations system except $Eq_{a^{out}}$ is different ($r \succeq FW^c(C_{a_5}) + 0$) from the one in equation system (1). By applying the diagnosis algorithms, the diagnosis that concerns the symptom marking M_{n_1} is illustrated in figure 4 while the diagnosis that concerns M_{n_2} is the $\overset{\cup}{\times}$ product of the diagnosis illustrated in figures 4 as the inequations system for symptom marking M_{n_2} contains one more red token in the activation output place a^{out}. In figure 4, we illustrate the diagnosis solving process in structured trees. The nodes represent the inequations needed to be solved and each leaf represents a diagnosis and the union of all leaves is a diagnosis solution.

$$\begin{cases} Eq_{a^{in}} : 0 \succeq (r - C_{a^{in}}) \times n_0 - C_{a^{in}} + b \\ Eq_{a_2} : 0 \succeq FW^c(C_{a^{in}}) - C_{a_2} \times 2 - C_{a_2} + FW^c(C_{a_4}) \times 2 + 0 \\ Eq_{a_3} : 0 \succeq EL^c(C_{a_2}, C_{d_2}) \times 2 - C_{a_3} \times 2 + 0 \\ Eq_{a_4} : 0 \succeq FW^c(C_{a_3}) \times 2 - C_{a_4} \times 2 + 0 \\ Eq_{a_5} : 0 \succeq EL^c(C_{a_2}, C_{d_2}) - C_{a_5} + 0 \\ Eq_{a^{out}} : * \succeq FW^c(C_{a_5}) + 0 \\ Eq_{d_0} : * \succeq (r - C_{d_0}) \times n_1 + b \\ Eq_{d_1} : * \succeq (r - C_{d_1}) \times n_2 + b \\ Eq_{d_2} : * \succeq FW^c(C_{d_0}) - C_{d_2} + (r - C_{d_2}) \times n_3 + b \\ Eq_{d_3} : * \succeq FW^c(C_{d_1}) + (EL^c(C_{d_4}) - C_{d_3}) \times 2 - C_{d_3} + b \\ Eq_{d_4} : * \succeq (EL^c(C_{d_2}) - C_{d_4}) \times 2 + (r - C_{d_3}) \times n_4 + b \\ Eq_{d_5} : r \succeq EL^c(C_{d_3}) - C_{d_5} + b \end{cases} \quad (1)$$

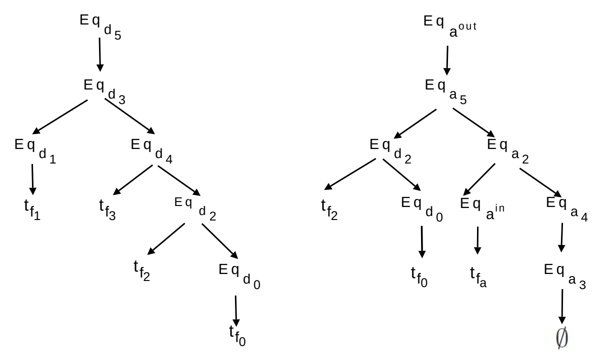

Fig. 4. Diag: M_{n_1} and Diag: $Eq_{a^{out}}$ in M_{n_2}

As a result, for symptom marking M_{n_1}, we have the diagnosis: D_1={{t_{f_0}}, {t_{f_1}}, {t_{f_2}}, {t_{f_3}}} represents 4 single faults. Either the input data fault d_0, or the input data fault d_1, or the transition fault on invoke activity S, or the transition fault on invoke activity O. Concerning the symptom marking M_{n_2}, the diagnosis is extended as: $D=D_1 \overset{\cup}{\times} D_2$, where D_2 concerns the red token in activation place a^{out}. As illustrated in figure 4 (right), $D_2=\{\{t_{f_a}\},\{t_{f_0}\},\{t_{f_2}\}\}$. So, we get diagnosis $D=\{\{t_{f_0}\}$, $\{t_{f_2}\}$, $\{t_{f_a},t_{f_1}\}$, $\{t_{f_a},t_{f_3}\}\}$, i.e., the fault is on input data place d_0, or on transition O, or on input activation place f_a and invoke activity C, or on input activation place f_a and invoke activity S. And these results can be used further for the study of diagnosability.

5. RELATED WORK AND CONCLUSION

A BPEL process can be considered as a **discrete event system** (DES). Automata, process algebra, and Petri nets are the most popular models. We refer the reader to Yan [2008] for the surveys of formal methods of Web services modeling. The major method for diagnosing a DES is trajectory unfolding. Unfolding method is used on the observable trajectory of system evolution to find the faulty states as the diagnosis. For example, Ye and Dague [2008] proposes a decentralized model-based diagnosis algorithm based on the PNs model (Li et al. [2007]) by inversely unfolding the trajectory. But in Ye and Dague [2008], local diagnoser does not support iteration in BPEL processes.

We can also adapt the *flightagent* example according to the modeling methods of Benveniste et al. [2003] by modeling the states of the BPEL service as places and activities as transitions. As this modeling approach loses the data dependency which cannot ensure the diagnosis is as minimal as ours. S.Genc and S.Lafortune [2005] models a modular interacting system as a set of place-bordered Petri nets and proposes a distributed online diagnosis which applies algebra calculations from the local models and the communicating messages between them. But the fact that S.Genc and S.Lafortune [2005] models the state of a model as a transition which causes the combinatorial explosion of the state space, and its simple Petri nets definition are too limited to deal with the data aspects.

There are some works that model the WS system with other types of models. In Console et al. [2002], a system is modeled with process algebra containing faulty behavior models. The diagnosis is done by comparing all possible action traces with the observations. All the faulty actions of the matched traces are the diagnosed faults. But Console et al. [2002] models and diagnosis the general WS applications but not a concrete WS specification language. Yan and Dague [2007] models BPEL services as enriched synchronized automata pieces and diagnose by trajectory reconstruction from observation while the algorithm is incapable for the control fault in the process.

This CPN modeling approach addresses diagnosis of semantic fault(s) of orchestrated Web services. The paper constructs a model for the faulty data and faulty activities in a BPEL process. We construct an inequations system for the diagnosis of a BPEL service. And a concrete inequations solving algorithm is proposed. The diagnosis takes advantage of the matrix calculation, which helps to improve the effectiveness of the diagnosis. The interpretation of happened (1) or not happened (0) status of the fault transitions avoids the unfolding of Petri nets. So the iterative structure in BPEL services does not increase the complexity of the diagnosis.

Our diagnosis approach can be easily extended into the distributed environments according to the approach proposed in S.Genc and S.Lafortune [2005] by defining a proper composition protocol of the CPNs. And we believe that the diagnosability analysis can also be done using algebra analysis based on the incidence matrix, which is another ongoing work.

REFERENCES

L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. T. Dupré. Enhancing web services with diagnostic capabilities. In *ECOWS*, pages 182–191, 2005.

A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *AC*, 48:714–727, 2003.

K. Boukadi, C. Ghedira, Z. Maamar, and H. Boucheneb. Specification and verification of views over composite web services using high level petri-nets. Technical report, 2006.

T. Chatain and C. Jard. Models for the supervision of web services orchestration with dynamic changes. In *AICT/SAPIR/ELETE*, pages 446–451. IEEE CS, 2005.

L. Console, C. Picardi, and M. Ribaudo. Process algebras for systems diagnosis. *AI*, 142(1):19–51, November 2002.

M. Diaz. *Les réseaux de Petri de haut niveau*. Hermes Science Publications, Paris, France, 2001.

K. Jensen. *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*. Springer, USA, 1997.

Y. Li, T. Melliti, and P. Dague. Modeling bpel ws for diagnosis: towards self-healing ws. In *WEBIST*, pages 795–803. IEEE C.S., 2007.

OASIS. Bpel 2.0 specification. `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html`, August 2006.

S.Genc and S.Lafortune. Distributed diagnosis of place-bordered petri nets. *ASE*, 4(2):206–219, April 2005.

W. Tan, Y. Fan, and M. Zhou. A petri net-based method for compatibility analysis and composition of web services in business process execution language. *ASE*, 6: 94–106, 2009.

Y. Yan. *Description Language and Formal Methods for Web Service Process Modeling*. M.E Sharpe Inc., Armonk USA, 2008.

Y. Yan and P. Dague. Modeling and diagnosing orchestrated web service process web services. In *ICWS IEEE International Conference*, pages 9–13. IEEE C.S., 2007.

L. Ye and P. Dague. Decentralized diagnosis for bpel web services (poster). In *WEBIST*, pages 283–287. INSTICC, 2008.

Z. Zhang, F. Hong, and H. Xiao. A colored petri net-based model for web service composition. *Journal of Shanghai University (English Edition)*, 105(4):323–329, 2008.