# A Decentralized Model-Based Diagnosis for BPEL services

Yingmin Li, Lina Ye, and Philippe Dague
LRI, Univ. Paris-Sud, CNRS, and INRIA Saclay-Île de France
Parc Club Orsay Université, 4 rue Jacques Monod, bât G, Orsay, F-91893, France
Email:⟨firstname⟩.⟨lastname⟩@lri.fr, Tel: (33) 172925993

Tarek Melliti
IBISC, Univ. d'Evry Val d'Essonne, CNRS,
Tour Evry 2, 523 place des Terrasses de l'Agora, Evry, F-91025, Franc
Tarek.Melliti@ibisc.fr

## Abstract

*The paper proposes a decentralized diagnosis approach for a set of choreographed BPEL Web services, where a local diagnoser is associated to each BPEL service and cooperates with a coordinator. The local diagnosis is based on a Colored Petri Nets model enriched with I/O data dependency relations represented with color propagation functions. By applying the multiset marking calculation equation, a diagnosis inequations system is constructed and solved to retrieve a local diagnosis. The coordinator updates the global diagnosis until reaching a final consistency.*[1]

## 1 Introduction

Self-healing software is one of the important challenges for Information Society Technologies research. Our paper proposes a decentralized diagnosis approach for BPEL 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? 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. These two kinds of faults are named as semantic faults.

### 1.1 Example: flight agency

Consider a cross-organization cooperation, which involves four partners: an end-user $User$, a Customer agency (flight agency) $C$, an airline company $A$, and a Bank center $B$. The three companies aim to offer a secure tickets buying service for $User$. $C$ $receive$s a request, which is composed of flight dates $d_1$, $d_2$ and departure and return cities $c_1$ and $c_2$, from $User$, and $invoke$s operation $o_1$ to get the cheapest flight from $A$. Next $C$ $invoke$s an operation $o_2$ to pay on $B$. Then in a $pick$ process, if $OnMessage$ of $C$ is "no credit", $C$ $invoke$s an operation $o_3$ to cancel the reservation on $A$ and should $reply$ to $User$ a failure message, else if $OnMessage$ of $C$ is a payment confirmation from $B$, $C$ should $reply$ a reservation information like $d_0$, $d_1$, $c_1$, $c_2$, airline name, flight number, price, etc. Partner

[1]The work 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).

$A$ $receive$s a request from $C$ to get the cheapest flight, $A$ $invoke$s a basic WS $BWS_1$ to get the cheapest available flight, and should $reply$ it to $C$. Then in a $pick$ process, if $OnMessage$ of $A$ is a payment confirmation from $B$, process finishes, else if $A$ is $OnMessage$ a cancel request from $C$, $A$ $invoke$s a basic WS $BWS_2$ to cancel the flight and terminate the process. $B$ $receive$s a payment order from $C$, and $invoke$s a basic WS $BWS_3$ to do deduction, then in a $switch$ process, if the deduction fails, $B$ $invoke$s an operation $o_4$ on one-way to send a "no credit" message to $C$, else $B$ $invoke$s operations $o_5$ and $o_6$ on one-way to send a confirmation message to $A$ and $C$.

Suppose $C$ is a French customer agent, while $A$ is an English airline company, and there is some semantic incompatibility between the two partners. A French reservation request like: reserve a round trip flight from Paris to London on March 4, 2009 (04/03/2009) and return 3 days later, on March 7, 2009 (07/03/2009) could be interpreted as reserving a round trip from Paris to London on April 3, 2009 and return 3 months later, on July 3, 2009. So $C$ will receive a wrong payment confirmation from $B$ and send it to $User$. In this case, an exception occurs on $User$ due to a semantic fault. This example will be used to illustrate each step of our work.

### 1.2 Global view of decentralized diagnosis

When an exception is thrown, the service that generates the data and all the services modify the data should be suspected. Whereas a current Web service exception can only report where the exception happens. Since the semantic faults are the most difficult and critical ones to diagnose, our approach focuses on determining the exact source faults that are possibly responsible for the exceptions.

A BPEL process can be considered as a Discrete Event System (DES) and a set of BPEL services in a choreographed environment can be modeled as a set of place-bordered Petri nets. We propose a decentralized online diagnosis architecture (in figure 1) in which, for each BPEL service $BPEL_i$, a local diagnoser $D_i$ is provided and cooperates with a coordinator $CO$. The coordinator contains a global diagnoser $D$ and a global choreographed model $Model$ which acknowledges the choreographed relations $CR$ between the interacting BPEL services. When an exception occurs on communicated $BPEL_i$, $D_i$ is triggered and then infers a local diagnosis result to $CO$ and that updates the status of $D$. If $D$ finds that $BPEL_j$ ($j \neq i$) is accused to be faulty, it triggers $D_j$. $D$ is updated continuously until arriving a final consistency.

To achieve a decentralized diagnosis in a choreographed scenario, we first introduce a local model for each BPEL service in section 2, where we introduce a CPN model and define its firing rules in section 2.1, translate the typical ba-

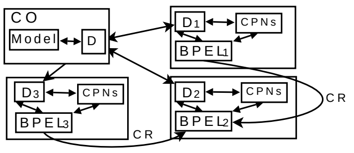


**Figure 1.** Diagnosis architecture

sic BPEL service activities and structural operators in to CPN models in section 2.2, especially, including the fault models locally. Next step, we construct the choreographed global model which includes the global fault model in section 3. The global and local diagnoses are presented in section 4. For the local diagnosis, an inequation diagnosis system is constructed and solved according to the algebra properties of the CPN model in 4.1. And we describe the decentralized diagnosis steps in 4.2. We compare the related work, and conclude in section 5.

## 2 Local model for one BPEL service

### 2.1 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 [6]. For two given value domains $D$, $D'$, we denote by $[D \to D']$ 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:* $\Sigma$ *is a set of colors (see [6]);* $\mathcal{X}$ *is set of variables that range over* $\Sigma$*;* $F$ *is a set of color functions,* $F \subseteq \bigcup_n[\Sigma^n \to \Sigma]$*;* $P$ *is a set of labeled places;* $T$ *is a set of labeled transitions, we denote* $Type : T' \to T''$ *with* $T', T'' \subset T$ *and* $T' \cap T''=\emptyset$ *is a type function of* $T$*;* $cd : P \to 2^{\Sigma}$*, is a function that associates to each place a color domain*[2]*;* $Pre, Post$ *: are forward and backward matrices such that* $Pre : P \times T \to \mathcal{M}^+(\Sigma \cup \mathcal{X})$*, are input arc expressions; and* $Post : P \times T \to \mathcal{M}^+(\mathcal{E})$*, 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

[2]In this definition, a transition has no color domain. This restriction will be explained in section 2.2.3.

$Var(e)$ to denote the set of variables which appear in $e$, and $Eval(e)$, the evaluation of $e$ in $\Sigma$.

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 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.* [3]

**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 $M[t\rangle^u M'$. A sequence of transitions $\delta \in T^*$ is defined as: $M[\delta\rangle M$ if $\delta$ is the empty sequence; $M[\omega t\rangle M'$ iff $\exists M''$ such that $M[\omega\rangle M''$ and $M''[t\rangle^u M'$.*

## 2.2 From BPEL to CPN model

There exist already many works dedicate to translate BPEL services into CPN model for composition verifying ([12]), supervising ([3]), analyzing ([5]) 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. In the following, we define how to translate the static and dynamic features into CPN models.

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

### 2.2.1 Translating static BPEL features to CPNs

**BPEL data variables and constants:** to catch maximally the dependency between data (variables, constants, etc.), we decompose the structured XML data types into $leaves$. Each leave $x_i$ is denoted by $(X, x_i)$ as a place in CPNs.

**Color Domain:** three colors are used: red (r), faulty data value; black (b), not faulty data value; and unknown color ($*$), correctness unknown 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 (proposed in [1]): forward ($FW$), the activity just copies the value from the input to the output; source ($SRC$), the output data is generated by the activity; and elaboration ($EL$), the output data is elaborated from a set of input data. Each data dependency relation is associated with a color propagation function to represent the data status production.

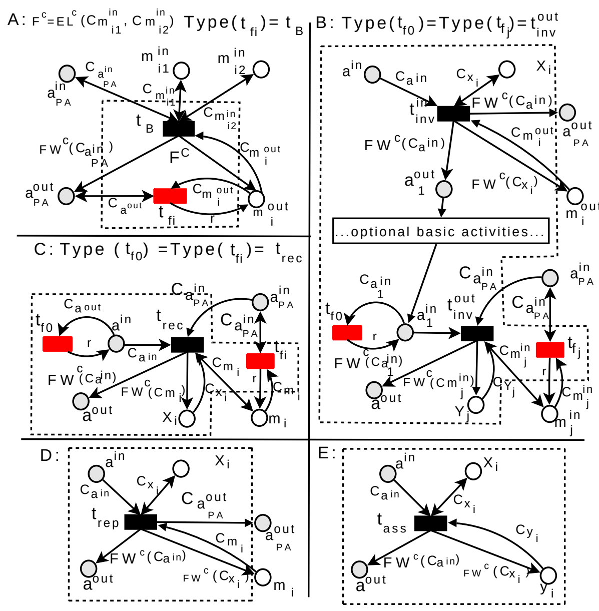
**Definition 7** *Given a data relations set $D = \{FW, SRC, EL\}$, $\forall d \in D$, the associated color propagation function $d^c$ is defined as: $\forall c, c' \in \Sigma, \forall \mathcal{C} \subseteq \Sigma$,*

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

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

### 2.2.2 Translation from basic WS to CPN

A basic WS is a program which publishes its invocation interface and can be remotely called by other WS. As it is called synchronously and cannot be decomposed, we model it as a CPN system, which has a transition, remote input/output activation places and a set of shared input/output data places (the local components are in the dotted line boxes). The data dependency between its input and output can be $FW$, $EL$, and/or $SRC$, which is optional offered by the developers. The CPN model of a basic WS contains a set of 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 the fault can be passed to its invoker, a BPEL process. We define the type of faulty transitions as $t_B$: $Type(t_{f_i}) = t_B$ (figure 2(A)).



A: $F^c=EL^c(Cm^{in}_{i1}, Cm^{in}_{i2})$ $Type(t_{fi})=t_B$
B: $Type(t_{f0})=Type(t_{fj})=t^{out}_{inv}$
C: $Type(t_{f0})=Type(t_{fi})=t_{rec}$
D:
E:
...optional basic activities...

**Figure 2.** CPNs of the basic activities

### 2.2.3 Basic activities translation

Due to space limitation, we model only four main basic activities ($Receive$,$Assign$,$Invoke$, and $Reply$) while the other similar activities can be easily translated in the same way (structural operators also).

**Receive**$(m, X)$ copies the values from a message $m$ to a local variable $X$. To separate the different faulty parts transmitted remotely, we add for each $m_i$ an internal transition (fault) before the receive transition as in figure 2(C). Data places $(m, m_i)$, $(x, x_i)$ are simplified as $m_i$ and $x_i$. A remote activation place $a^{in}_{PA}$ is added to allow the activation control from partner $PA$ to $Receive$.

Two kinds of fault transitions are modeled: $t_{f_0}$ models the remote control fault; and $t_{f_i}$ models the remote data fault. Both their types are defined as $t_{rec}$: $Type(t_{f_0}) = Type(t_{f_i}) = t_{rec}$. The execution (occurrence) of $t_{f_i}$ is triggered by the consummation of the token in the remote input activation place $a^{in}_{PA}$. 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 CPN, we add an arc from the output place $x_i$ to the receive transition $t_{rec}$. Similar activity: $OnMessage$.

**Invoke**$(X[, Y])$ calls another Web service, either a basic or composite one. It takes the value of the variable $X$, sends a remote request to its partner, synchronously or asynchronously waits for the response message, stores it in the variable $Y$, or gets the response by $Receive(m, Y)$. As $Y$ can be infected by external faulty WS which is unobservable, we introduce a series of unobservable faulty transitions between the $t^{in}_{inv}$ and $t^{out}_{inv}$ transitions to model the faults caused by external WS. $Invoke$ can be asynchronous call and only sends the request $X$. In this case, it has only the $t^{in}_{inv}$ transition (figure 2(B)).

**Reply**$(Y, m)$ copies values from a variable $Y$ to a message $m$ for returning the response (figure 2(D)). There is no fault model in its CPN model. Note that $reply$ does not affect the correctness of remote control, so $f^c$ on the arc $(t_{rep}, a^{out}_{PA})$ is $FW^c(a^{in}_{PA})$.

**Assign**$(X, Y)$ reorganizes local variable parts to compose a new one. So its model does not contain fault transition, remote activation, or shared data places (figure 2(E)). 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 I/O local and remote activation places.

### 2.2.4 Structured operators translation

**Sequence operator** $sequence(S_1, S_2)$ connects different activities with and the occurrence execution order. So we can generate the resulting sequence CPN by merging the local intermediate output and input activation places of contractive CPNs (in figure 3(a)).

**Conditional operator** $Switch(\{(con_i(\overline{X_i}, \overline{V_i}), S_i)\}_{i\in I})$ 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 $S_i$, a transition $con_i$ is added to generate an activation place $a^{in}_i$ which is **elaborate** from the common activation input place of $Switch$, $\overline{X_i}$, and $\overline{V_i}$. So the faults in the data places $\overline{X_i}$, $\overline{V_i}$ can be transmitted to subprocess activation control, which allows the diagnosis of the control fault in a BPEL process. At the end of the $Switch$ process, a new $a^{out}$ replaces all the $a^{out_i}$ of subprocess $S_i$ (in figure 3(c)). Similar operators: $Scope$ together with the compensation handlers, event handlers, and fault handlers.

**Iterative operator** $while(con(\overline{X}, \overline{V}), S_1)$ iterates the activity $S_1$ execution until the breaking off of the conditions $con(\overline{X}, \overline{V})$. $While$ is similar to $Switch$ but in $While$, the $a^{out}$ of the iterative subprocess is also $a^{in}$ of $t_{con}$ (in figure 3(b)). Similar operators: $Link$ with its "transitionCondition". Similar operators: $RepeatUntil$, $ForEach$.

**Message triggering operator** $Pick(\{M_i, S_i\}_{i\in I})$ triggers one subprocess $S_i$ by the arriving (represented as the remote activation place $a^{in}_{PA_i}$) of a message $OnMessage(M_i)$ from partner $PA_i$. So $Pick$ operator is a combination of a set of $OnMessage$ activities (in figure 3(d)).

### 2.2.5 Some remarks on the BPEL model net

**Observable vs unobservable transitions**: we divide $T$ into observable transitions $T_{obs}$ and fault transitions $T_F$ ($T =$

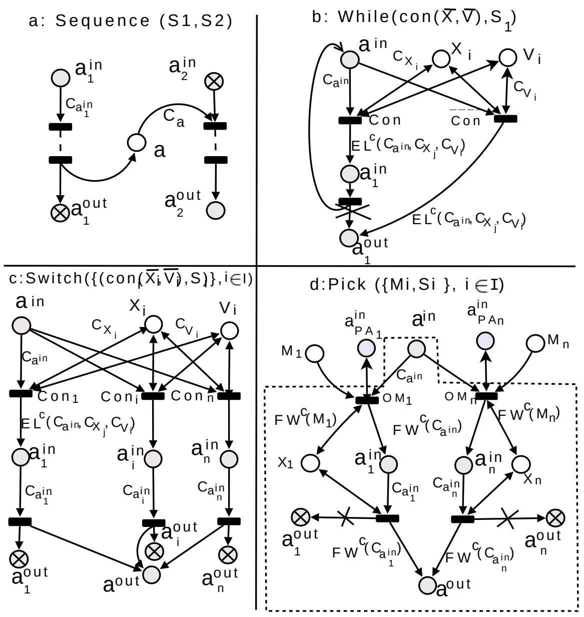


**Figure 3.** CPNs of the structural operators

$T_{obs} \cup F$ and $T_{obs} \cap T_F = \emptyset$) and define a type function over faulty transition to observable ones $Type : T_F \rightarrow T_{obs}$.

**Initial and final marking configuration**: obtained by marking $P$ as in table 2.2.5.

**Table 1.** $M_0$ and $M_n$ of CPNs

| | $CP^1$ | $DP^2$ | $AP_0{}^3$ | $AP_0{}^4$ | $AP^0_{PA_0}{}^5$ | $AP^0_{PA_i}{}^6$ |
|---|---|---|---|---|---|---|
| $M_0$ | $*$ | $b$ | $b$ | 0 | $b$ | 0 |
| $M_n$ | $*$ | $x^7$ | 0 | $x^7$ | 0 | $x^7$ |

[1] Constant places; [2] Variable places;
[3] First local activation places; [4] Local activation places;
[5] First remote activation places;
[6] Remote activation places;
[7] $x$=$r$ if $p$ is within fault trace, $x$=$b$ if verified by external checkers (e.g., user, developer), $x$=$*$ otherwise.

**One-boundness:** the resulted CPNs are one-bounded (or safe, means one place can at most contain one token), which is guaranteed by the fact that a BPEL process does not allow a subprocess call that can lead to more than one token production in the activation and data places.

### 2.2.6 Example (cont.): flight agency

In figure 4, a CPN model of partner $C$ is presented together with its interacting relations (bolded arrows) with other partners (data places for each transition are simplified as one input and one output for the sake of space limit). All the color variables $C_p$ of a place $p$ are omitted and the color propagation functions are listed. A symptom (red token) occurs on $User$ to represent a faulty reservation which shows an unreasonable price.

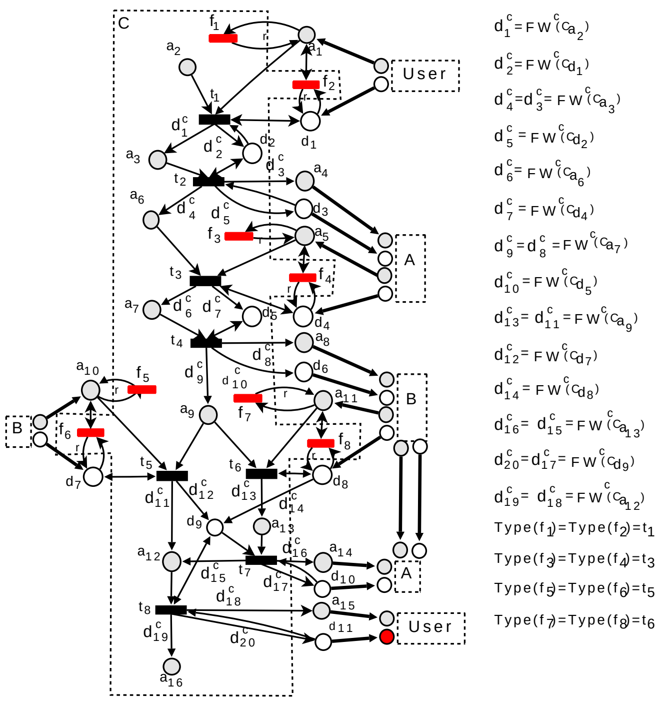


**Figure 4.** CPNs model of partner $C$

## 3 Fault model of a choreographed set of WSs

Each partner $N_i$ involved in a choreography scenario provides a BPEL process describing: (i) its partners $\{N_j\}(j \neq i)$ and (ii) its interface (local operations like $Invoke$, $Receive$, and $Reply$). A choreographed relation between two partners is represented by two sets of places which correspond to a share messages and their corresponding remote control places.

**Definition 8** *let $P_i^s$ denote the subset of places $P_i$ on $N_i$ which are shared with another $N_j$ ($j \neq i$). A choreographed relation $\mathbb{CR}$ is a set of functions $F_{ij} : A^{out}_{PA_i} \times P_i^s \rightarrow A^{in}_{PA_j} \times P_j^s\}$. $\forall a \in A^{out}_{PA_i}$, $\forall a' \in A^{in}_{PA_i}$, $p \in P_i^r$, $F_{ij}(a,p)$=$(a',p')$ and $F^{-1}_{ij}(a',p')$=$(a,p)$.*

**Definition 9** *A choreographed model is a tuple $CDM$= $\langle \bigcup\limits_{i \in I} N_i$, $\mathbb{CR}\rangle$, where $N_i$ is a CPN model of a partner BPEL service, and $\mathbb{CR}$ is a choreographed relation.*

### 3.1 Example (cont.): flight agency

In figure 4, the interacting places are: $(a_i, d_j)$ with $(i,j)$=(1,1),(4,3),(5,4),(8,6),(10,7),(11,8),(14,10),(15,11).

We simply denote the shared places on the corresponding partners as $a'_i$ and $d'_j$. So the choreographed relation of partner $A$ to $B$ and $C$ is: $\mathbb{CR}_A = \bigcup_{\alpha \in \{B,C\}} F_{A\alpha}$. The global model of the flight agent services $fa$ set is: $CMD_{fa} = \langle N_A \cup N_B \cup N_C, \mathbb{CR}_A \cup \mathbb{CR}_B \cup \mathbb{CR}_C \rangle$.

## 4 Diagnosis of Decentralized BPEL services

### 4.1 Local Diagnosis for one BPEL service

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

**Definition 11** *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; $\delta_o$ is an observable trace $\delta_o \in (T_{obs})^*$; $\hat{M}$ is a symptom marking.*

To solve a diagnosis problem, we introduce a covering relation:

**Definition 12** *A covering relation $\preceq$ between colors $\Sigma = \{r, b, *\}$ 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 13** *Let $\mathcal{D} =< N, \delta_o, \hat{M} >$ be a diagnosis problem, a diagnosis $S$ is a non empty subset of faults set ($T_F$), ($S \subseteq T_F$) such that: $M_0 + C \times \overrightarrow{\delta} \preceq \hat{M}$ with $\overrightarrow{\delta}$ 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 number of occurrences of $t$ in $\delta_o$; $\forall f \in S$, $\overrightarrow{\delta}(f) = 1$; $\forall f \in (T_F \setminus S)$, $\overrightarrow{\delta}(f) = 0$.*

We restrict the value of a fault transition to 1 because once fault occurs, the fault will be persistent.

**Definition 14** *Let $\mathcal{D} = \langle N, \delta_o, \hat{M} \rangle$ be a diagnosis problem and $S$ be a diagnosis, $S$ is minimal iff $\forall S' \subset S$, $S'$ is not a diagnosis. The diagnosis solution $\mathcal{DS} \subseteq 2^{T_F}$ is the set of all possible minimal diagnoses.*

#### 4.1.1 Diagnosis of CPN by inequations system solving

Let $\mathcal{D} = \langle N, \delta_o, \hat{M} \rangle$ be a diagnosis problem and let $n_i$ be variables ranging over $\{0, 1\}$, we construct the characteristic vector $\delta$ as follows: $\forall t \in T_{obs}$, i) $\overrightarrow{\delta}(t)$=$\overrightarrow{\delta_o}(t)$; ii) $\forall f_i \in T_F \wedge \overrightarrow{\delta_o}(Type(f_i)) \neq 0$, $\overrightarrow{\delta}(f_i)$=$n_i$; iii) $\forall f \in T_F \wedge \overrightarrow{\delta_o}(Type(f))$=0, $\overrightarrow{\delta}(f)$=0;

We can then construct an inequations system (one inequation for each place) for the diagnosis problem as follows: $Q_{\hat{M}}$=$Eq_{p_i}$ : $\hat{M}(p_i) \succeq M_0(p_i) + C(p_i, .)\overrightarrow{\delta}$.

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: i) $Q^r_{\hat{M}}$=$\{Eq_p | l(Eq_p)$=$r\}$; ii) $Q^b_{\hat{M}}$=$\{Eq_p | l(Eq_p)$=$b\}$; iii) $Q^*_{\hat{M}}$=$\{Eq_p | l(Eq_p)$=$* \vee l(Eq_p)$=$0\}$.

The diagnosis is obtained by equating the left and the right parts of the $Q^r_{\hat{M}}$ equations. In the following, we give first the solution of an inequation and then the solution of an inequations system.

**One inequation $Q^r_{\hat{M}}$ solving**
The inequation with the left red part is a multi set over terms that can be composed by color functions, constants, and the corresponding place variables (could be with negative coefficients). The solving approach (algorithm 1) is, for each inequation corresponding to one symptom place $p$, to balance out the negative items with the positive constant items in the right part, initiate the red tokens coefficients $n_i$ as 1 (fault occurs), and keep the unknown color functions for further recursive solving (algorithm 2) until arriving to a choreographed relation.

#### 4.1.2 Example (cont.): partner $C$ of flight agency

The incidence matrix, with a size $27 \times 16$ can be constructed by applying the $Post$-$Pre$. Now suppose we get a series of observed activities $\delta_0$: $Receive$, $Invoke$, $Invoke$, $Pick$, $OnMessage$, $Reply$ (the corresponding transitions in CPN model are $t_1$, $t_2$, $t_3$, $t_4$, $t_5$, $t_8$), which means the payment confirmation branch is executed. Given an initial marking $M_0$ =($a_1$ $a_2$ $a_3$ $\cdots$ $a_{16}$ $d_1$ $\ldots d_{11}$)=($b$ $b$ 0 $\cdots$ 0 $b$ $\cdots$ $b$), the final marking is $M_n$ =($a_1$ $\cdots$ $a_{14}$ $a_{15}$ $a_{16}$ $d_1$ $\cdots$ $d_{10}$ $d_{11}$)=(0 $\cdots$ 0 $b$ $b$ $*$ $\cdots$ $*$ $r$). The characteristic vector can be constructed as: $\overrightarrow{\delta}^T$: ($t_1$ $\cdots$ $t_6$ $t_7$ $t_8$ $f_1$ $\cdots$ $f_6$ $f_7$ $f_8$)=(1 $\cdots$ 1 0 0 $n_1$ $\cdots$ $n_6$ 0 0). An inequations system concerning

**Algorithm 1** Partially solving one inequation: $solvAnEqu(Eq_p)$

```
Input: $Eq_p$: a $Q^r_{\hat{M}}$ inequation which 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 coefficients which mean their corresponding fault transitions are executed;}
1: $C^r_p = \emptyset$; $N^r_p = \emptyset$;
2: ForEach $n_i \times c_i \in r(Eq_p)^+ = \sum_{i \in I} n_i \times c_i$ do
3:     if $n_i$ is not a constant and $c_i = r$ then
4:         $N^r_p = N^r_p \cup \{f_i\}$; {records $t_{f_i}$ in $N^r_p$}
5:     else if $c_i$ is a color function concerning place $p'$ then
6:         $C^r_p = C^r_p \cup \{c_{p'}\}$;{records possible fault place $c_{p'}$ for further solving}
7:     else if $c_i$ is a color propagation function $d^c_i$ then
8:         $C^r_p = \{C^r_p\} \cup \{c_{p_i} \in Var(c_i)\}$;{records the input places of $c_i$ for further solving}
9: return $< C^r_p, N^r_p >$;
```

partner $C$ can be got by applying definition 6 in equation 4.1.2 (the inequation $Eq_{d_{11}}$ in a gray box is symptom). By solving the inequations system, a diagnosis on partner $C$ is got: $n_6 > 0$, which means place $d_7$ is faulty, so we can accuse partner $B$ to be faulty. In the following sections, we see how to diagnose this fault in a choreographed scenario.

$$\begin{cases} Eq_{a_1} : 0 \succeq (r - C_{a_1}) \times n_1 - C_{a_1} \times 1 + b \cdots \\ Eq_{a_{16}} : b \succeq FW^c(C_{a_{12}}) \times 1 + 0 \cdots \\ Eq_{d_2} : * \succeq (FW^c(C_{d_1}) - C_{d_2}) \times 1 + b \cdots \\ Eq_{d_{11}} : r \succeq FW^c(C_{d_9}) + b \end{cases}$$

**Algorithm 2** Completely solving a $Q^r_{\hat{M}}$ inequation: $CDS(Q_{\hat{M}}, Eq_p)$

```
Input: $Q_{\hat{M}} = Q^r_{\hat{M}} \cup Q^b_{\hat{M}} \cup Q^*_{\hat{M}}$: the inequations system ;
    $Eq_p$: a $Q^r_{\hat{M}}$ inequation;
Output: $S_p$: a diagnosis solution concerning a symptom place $p$;
1: $S_p = \emptyset$;
2: $< C^r_p, N^r_p >= solvAnEqu(Eq_p)$;{get the first back reasoning result, $C^r_p$ need to be resolve further}
3: $S_p = S_p \cup N^r_p$;
4: ForEach $c_{p'} \in C^r_p$ do
5:     if $\exists Eq_{p'} \in Q^*_{\hat{M}}$ then
6:         if $l(Eq_{p'}) = *$ then
7:             $S_p = S_p \cup CDS(Q^r_{\hat{M}} \cup \{r \succeq r(Eq_{p'})\} \cup (Q^b_{\hat{M}} \cup Q^*_{\hat{M}}) \backslash \{Eq_p, Eq_{p'}\}, r \succeq r(Eq_{p'}))$;{evaluates $l(Eq_{p'})$ as $r$, reconstructs the inequations system and recursively back reasoning}
8:         else if $l(Eq_{p'}) = 0$ then
               $S_p = S_p \cup CDS(Q^r_{\hat{M}} \cup \{r \succeq r(Eq_{p'}) + c_{p'}\} \cup (Q^b_{\hat{M}} \cup Q^*_{\hat{M}}) \setminus \{Eq_p, Eq_{p'}\}, r \succeq r(Eq_{p'}) + c_{p'})$;{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}
9: return $S_p$;
```

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

By solving each inequation in $Q^r_{\hat{M}}$, we get the diagnosis for the inequations system $Q_{\hat{M}}$ (see algorithm 3). The union set of all the $S_p$ is the diagnosis solution for the whole symptom marking which corresponds to multiple symptoms.

## 4.2 Protocol for global diagnosis

In the decentralized diagnosis architecture, each local diagnoser can interact both with its associated Web service and with the coordinator, while the coordinator can interact only with local diagnosers. The coordinator $CO$ contains a global model $CDM$ and an initially empty global diagnoser $D$. This decentralized diagnosis starts from a local diagnoser $LD_i$ which is triggered by $BPEL_i$. $D$ is first extended by $LD_i$. $D$ keeps the fault transitions $f_i$ and invokes other local diagnoser for further explanation of faulty places $p_i$ until a local diagnoser returns null (see algorithm 3).

**Algorithm 3** Local diagnosis for $Q_{\hat{M}}$: $LDS(\{p_i\})$

```
Input: $\{p_i\}$: a set of faulty places in local CPN model;
Output: $D$: a local diagnosis solution;
1: $D$=$\emptyset$;
2: $Q_{\hat{M}}$=$Construct(\{p_i\})$;{$Construct$ constructs an inequations system $Q_{\hat{M}}$ where $Q^r_{\hat{M}}$=$\{l(Eq_{p_i}) = r\}$}
3: ForEach $Eq_p \in Q^r_{\hat{M}}$ do
4:     $S_p$=$CDS(Q_{\hat{M}}, Eq_p)$;{resolve each inequation in $Q^r_{\hat{M}}$ by back reasoning}
5:     $Q_{\hat{M}}$=$Q_{\hat{M}} \cap S_p$;
6:     $D$=$D \overset{\cup}{\times} S_p$;[4]
7: return $D$;
```

## 4.3 Example (cont.): flight agency

Concerning the flight agency (figure 4), the diagnosis process starts on partner $C$ which is triggered by a wrong reservation exception. A local diagnosis result $D_{C_1}$=$\{\{\langle C, d_7 \rangle\}\}$ is sent by $C$ to the global diagoser $D$. $D$ is then extended as $\{\{\langle C, d_7 \rangle\}\}$ and according to the global model known by the coordinator, $\{\{\langle B, d'_7 \rangle\}\}$, the corresponding fault signature on partner $B$, is generated and sent to $B$. Next $D$ receives a local diagnosis $D_{B_1}$=$\{\{\langle B, d'_6 \rangle\}\}$ and $\{\langle B, d'_7 \rangle\}$ in $D$ is replaced by $\{\langle B, d'_6 \rangle\}$. The process continues and $D$ is extended in turn by the local diagnosis results $\{\{\langle A, d'_4 \rangle\}\}$, then $\{\{\langle A, d'_3 \rangle\}\}$, $BWS_1$, $\{\{\langle A, d_1 \rangle\}\}$, and $\{\{\langle A, d'_1 \rangle\}\}$ ($\{\{\langle User, d'_1 \rangle\}\}$ can be rejected by $user$). So the final diagnosis result as $D$=$\{\{\langle BWS_1, f_t \rangle\}\}$, where $D$ contains

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

one minimal diagnosis and $\{\langle BWS_1, f_t\rangle\}$, which indicates the fault come from the basic WS $BWS_1$ on partner A.

**Algorithm 4** Global diagnosis solution $GDS$

```
Input: $D$=$\bigcup\{\{\bigcup_{j\in N}\langle D_i,p_j\rangle, \bigcup_{k\in N}\langle D_i,f_k\rangle\}\}$: is a set of minimal
    diagnosis on $D_i$, $p_j$ is a suspended place on $D_i$, and $f_k$ is a
    faulty transition on $D_i$;
Output: $D$: the global minimal diagnosis set;
 1: $stop$=$fault$;{A flag to exit the diagnosis process}
 2: while Not $Stop$ do
 3:     ForEach $ld \in D$ do
 4:         ForEach $e \in ld$ do
 5:             $ld'$=$ld' \cup F^{-1}(e)$;{$F \in \mathbb{CR}$, while $A_{PA}^{in}$ and $A_{PA}^{out}$
                are omitted}
 6:         ForEach $e' \in ld'$ do
 7:             ForEach $\langle D_m, p_n\rangle \in e'$ do
 8:                 $ld'$=$ld' \overset{\cup}{\times} LDS_m(\{p_n\})$;
 9:         $D'$=$D\backslash\{ld\} \cup \{ld'\}$;
10:     if $D'$=$D$ then
11:         $stop$=$true$;
12:     else
13:         $D$=$D'$;
14: return $D$;
```

## 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 DES models. We refer the reader to [9] for the surveys of formal methods of Web services modeling. The major method for diagnosing a DES is trajectory unfolding, which is used on the observable trajectory of system evolution to find the faulty states as the diagnosis. For example, [11] proposes a decentralized model-based diagnosis algorithm based on the PNs model ([7]) by unfolding the trajectory backword. But in [11], local diagnoser does not support iteration in BPEL processes.

We can also adapt the $FlightAgent$ example according to the modeling methods of [2] 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. [8] 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 when applying [8] on the $FlightAgent$ example gets the explosion of the state space because the partition of the variables and messages into subtle parts, 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 [4], a system is modeled with process algebra which contains faulty behavior models. The diagnosis is retrieved by comparing all possible action traces with the observations. All the faulty actions of the matched traces are the diagnosed faults. But [4] models and diagnosis the general WS applications in stead of a concrete WS specification language. [10] 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.

An implementation of translating BPEL into CPNs based on the implementation of [7] is under developing in order to proof and compare the diagnosis minimality and correctness with other diagnosis approaches. Meanwhile we are studying a more general model for diagnosing various DES based on the inequations system solving approaches. Our diagnosis approach can be easily extended into the distributed environments according to the approach proposed in [8] 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

[1] 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.

[2] A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. on Automatic Control*, 48:714–727, 2003.

[3] 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 Press, 2005.

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

[5] C. O. et al. Formal semantics and analysis of control flow in ws-bpel. *Science of Computer Programming*, 67(2-3):162–198, July 2007.

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

[7] 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.

[8] S.Genc and S.Lafortune. Distributed diagnosis of place-bordered petri nets. *Automation Science and Engineering, IEEE Transactions*, 4(2):206–219, April 2005.

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

[10] 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.

[11] L. Ye and P. Dague. Decentralized diagnosis for bpel web services. In *WEBIST*, pages 283–287. INSTICC, 2008.

[12] Z. ZhaoLi, H. Fan, and X. HaiJun. A colored petri net-based model for web service composition. *Journal of Shanghai University (English Edition)*, 105(4):323–329, 2008.