Journal of Theoretical and Applied Information Technology
20th February 2013. Vol. 48 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

BPNDIAG: BPN-BASED FAULT DIAGNOSIS FOR BPEL PROCESS

ZHICHUN JIA, RONG CHEN
School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

ABSTRACT

Web Service is becoming an important and popular distributed computing model for providing on-demand service sharing on Internet. One of the difficult challenges in web service is how to deliver the reliable service application over unreliable web services around the world. To improve the automated diagnosis capability for web service, we completely model the BPEL process as a BPN model and propose a BPN model-based diagnosis method. Considering the activity dependency relations in BPN model, our diagnosis method is able to exactly and quickly localize the faulty activity. Finally, we conduct the simulation experiments to evaluate our method. The experimental results show that our method is effective in fault diagnosis for BPEL process.

Keywords: *Web Service, Fault Diagnosis, BPEL Process, Petri Nets, Diagnosis Probability*

1. INTRODUCTION

As a popular distributed computing model, web service makes software even more attractive due to it provides a lot of benefits like scalability, ubiquitous network access. However, along with these benefits, web service also raises some concerns especially how to build high-reliable service applications in the large-scale and complex computing environment. With the growing worldwide acceptance of the standard service composition language BPEL (Business Process Execution Language), how to detect and localize faulty activities in BPEL becomes a critical issue.

According to observed and thrown exceptions during the service execution, the goal of fault diagnosis is to timely and exactly detect abnormal activities, explain the faulty reason and bring the process back to a normal, safe, operating state. In 2005, Ardissono et al. [1] firstly applied model-based diagnosis (MBD) to web services composition. In recent years, many researchers proposed their fault diagnosis methods for web service [2-5]. Some researchers used automaton model [6], some used Petri nets model [7, 8], and others used probabilistic graphical model [9]. From methodology point view, some methods belong to exception handling in the field of software engineering; some others belong to MBD or statistical anomaly detection in the field of artificial intelligence (AI). However, these methods aren't adequate to meet the growing requirements of business processes for a diagnostic system.

To improve the diagnostic accuracy and efficiency, we propose a BPN model-based diagnosis method to diagnose faults in BPEL process. We firstly define the Petri nets-based process model for each activity in BPEL (BPN model). Then, we build the diagnosis model which defines the rules of diagnosis and faulty types. According to these rules and faulty types, we can quickly infer that which activity is fault. Moreover, to raise diagnostic efficiency, we first diagnose the more suspicious activity by computing the diagnosis probability of activities. Finally, we conduct the simulation experiments to evaluate the accuracy and efficiency of our method. The experimental results show that our method is more accuracy and costs less time than an existing MBD method.

The rest of paper is organized as follows. Section 2 introduces BPEL and Petri nets. Section 3 presents the BPN model. Section 4 proposes our diagnosis model and diagnosis method. Section 5 evaluates the experimental results. Section 6 discusses related work. Finally, we conduct the conclusion in Section 7.

2. PRELIMINARY

In this section, we simply introduce BPEL and Petri nets to be used in this paper.

2.1 BPEL

BPEL for web services is a standard executable language for specifying actions within business process based on composite web services. BPEL gives the formal specification for defining business

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

process behavior and business interaction protocols. It provides two kinds of activities for describing the internal behavior: basic activity and structured activity. Structured activities prescribe the order in which a collection of activities take place. They describe how a business process is created by composing the basic activities [10].

The set of basic activities includes:

- receive: waits for a matching message to arrive
- invoke: invokes an operation on a port offered by a partner
- reply: replies a message to a partner
- assign: copies to target variables with new data
- wait: waits for a given time period or until a certain time has passed
- throw: signals a fault
- empty: does nothing

The set of structured activities includes:

- sequence: defines a collection of activities to be performed sequentially
- switch: selects exactly one branch of activity by a condition it holds true
- while: defines loop execution of an activity until the condition has been met
- flow: specifies some activities to be performed concurrently
- pick: selects an activity path depending either on an occurring event or timeout

2.2 Petri nets

Petri nets are a basic modeling tool of parallel and distributed systems. They originated from Carl Adam Petri's dissertation in 1962 for the purpose of describing chemical processes [11]. The basic idea is to describe state changes in a system with transitions. The choice of Petri nets for modeling BPEL is motivated by following reasons: (1) the formal semantics is essential for representing the business process of web services compositions; (2) the causality between places and transition will help analysis the relationship between input, output and activities; (3) the graphical representation helps understand the whole model.

Petri nets contain places and transitions that may be connected by directed arcs. Transitions symbolize actions; places symbolize states or conditions that need to be met before an action can be carried out. Places may contain tokens that may move to other places by executing actions.

Definition 1 a Petri Net is a 3-tuple, $N=(P,T,F)$ where:

- P is a finite set of places;
- *T is a finite set of transitions;*
- $F \subseteq (P\times T)\bigcup(T\times P)$ *is the flow relation;*
- $P\bigcap T=\phi$ *and* $P\bigcup T\neq\phi$*;*
- $dom(F)\bigcup cod(F)=P\bigcup T$ *;*
- $dom(F)=\{x\mid\exists y:(x,y)\in F\}$*;*
- $cod(F)=\{y\mid\exists x:(x,y)\in F\}$.

Definition 2 the remarks of Petri Net $N=(P,T,F)$ are:

- *If* $(p,t)\in F$ *for a transition t and a place p, then p is an input place of t;*
- *If* $(t,p)\in F$ *for a transition t and a place p, then p is an output place of t;*
- *Let* $x,y\in P\bigcup T$*, the set* $^{\bullet}y=\{x\mid(x,y)\in F\}$ *is called the pre-set of y and the set* $x^{\bullet}=\{y\mid(x,y)\in F\}$ *is the post-set of x;*
- $\overset{\bullet}{x}=\{x\}$.

3. MODELING BPEL PROCESS

In this section, we model each BPEL activity as a BPN model which is an extended model of Petri nets. First, let us briefly give some remarks:

- Each activity and each BPN both start from a starting place and end in an ending place, and the pre-set of starting place and post-set of ending place are empty sets.
- Each place includes a set of variables seen as a token set. The token set moves to other places by transition and their values are changed by operation in the transition.
- Each transition includes an operation which represents the concept of operation in BPEL. An operation incorporates operation name, port name and service name. The conditions and expressions in activities are viewed as computing operations which obtain weight '*t*' or '*f*' in transition and select the next places by weight values.

Definition 3 a BPN model is $BPN=(p_s,p_e,P,T,F,W,V,OP)$, where:

- p_s *is the starting place of BPN,* p_e *is the ending place of BPN,* $p_s\in P\wedge p_e\in P$
- $P=\{p_1,p_2,\cdots,p_m\}$ is a finite set of places;
- $T=\{t_1,t_2,\cdots,t_r\}$ is a finite set of transitions, and $\forall t\in T,{}^{\bullet}t\neq\varphi\wedge t^{\bullet}\neq\varphi$;
- $F\subseteq(P\times T)\bigcup(T\times P)$ is the flow relation;
- $W:F\rightarrow\{t,f,\phi\}$ is the arc weight mapping, the default is ϕ ;
- $V=\{v_1,v_2,\cdots,v_k\}$ is a finite set of variables, $\forall 1\le i\le k,v_i=(vname,vtype,vvalue,Part)$, *vname* denotes the name of v_i , *vtype* denotes the type of v_i , *vvalue* denotes the value of v_i ;

Journal of Theoretical and Applied Information Technology
20th February 2013. Vol. 48 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- $Part = \{part_1, part_2, \cdots, part_l\}$, $\forall 1 \le i \le l$, $part_i = (name_i, type_i, value_i)$;
- l is the total count of parts, $name_i$ denotes the name of $part_i$, $type_i$ denotes the type of $part_i$, $value_i$ denotes the value of $part_i$;
- $OP = \{op_1, op_2, \cdots, op_n\}$ is a finite set of operations and $\forall 1 \le i \le n, op_i = (opn, port, service)$, where opn denotes the operation name, $port$ denotes the port name including the operation opn, $service$ denotes the service name including the port $port$;
- $\forall 1 \le i \le m \,\&\, \forall 1 \le j \le k$, $p_i.v_j$ denotes the variable v_j in place p_i ;
- $\forall 1 \le i \le r$, $t_i.op$ denotes the operation op in transition t_i .

3.1 Modeling basic activities

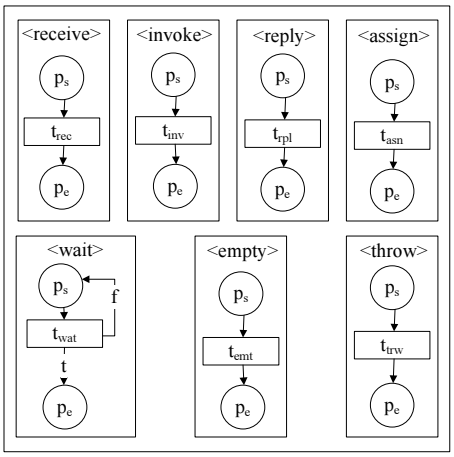

Figure 1: BPN Model Of Basic Activities

(1) receive

$$BPN_{rec} = (\{p_s, p_e\}, \{t_{rec}\}, \{(p_s, t_{rec}), (t_{rec}, p_e)\}, \{\phi\}, \{v\}, \{op_{rec}\})$$

Receive activity is activated when a message is received by operation op_{rec} , and assign the value of message to variable v which is received by place p_e .

(2) invoke

$$BPN_{inv} = (\{p_s, p_e\}, \{t_{inv}\}, \{(p_s, t_{inv}), (t_{inv}, p_e)\}, \{\phi\}, \{v_i, v_o\}, \{op_{inv}\})$$

An invoke operation op_{inv} can be a synchronous request/response or an asynchronous one-way operation. An asynchronous invocation only requires input variable v_i because it doesn't expect a response as a part of invocation, while a synchronous invocation requires an input variable v_i and an output variable v_o . Here, $v_i.vtype = v_o.vtype$.

(3) reply

$$BPN_{rpl} = (\{p_s, p_e\}, \{t_{rpl}\}, \{(p_s, t_{rpl}), (t_{rpl}, p_e)\}, \{\phi\}, \{v\}, \{op_{rpl}\})$$

A reply activity is used to send a response to a request previously accepted by a receive activity. Such responses are only meaningful for synchronous interactions. The variable v is to be used to construct the output message.

(4) assign

$$BPN_{asn} = (\{p_s, p_e\}, \{t_{asn}\}, \{(p_s, t_{asn}), (t_{asn}, p_e)\}, \{\phi\}, \{V_i, V_o\}, \{copy(V_i, V_o)\})$$

The assign activity can be used to copy message from a set of input variable V_i to a set of output variable V_o by operation copy.

(5) wait

$$BPN_{wat} = (\{p_s, p_e\}, \{t_{wat}\}, \{(p_s, t_{wat}), (t_{wat}, p_e), (t_{wat}, p_s)\}, \{t, f\}, \{\phi\}, \{cnd\})$$

The wait activity specifies a delay for a certain period of time or until a certain deadline is reached. In our model, we don't consider time. Hence, the notion of delay is not considered, it is viewed as a condition *cnd*. When the condition is met (*W=t*), the activity moves to the next place p_e , otherwise (*W=f*) it will wait to be met in place p_s .

(6) throw

$$BPN_{trw} = (\{p_s, p_e\}, \{t_{trw}\}, \{(p_s, t_{trw}), (t_{trw}, p_e)\}, \{\phi\}, \{V_f\}, \{throw(V_f)\})$$

The throw activity can be used when it needs to signal an internal fault explicitly. It is required to provide such a name $V_f.vname$ for the fault and can optionally provide a data $V_f.value$ that provides further information about the fault. A fault handler can use such data to analyze and handle the fault and also to populate any fault messages that need to be sent to other services.

(7) empty

$$BPN_{emt} = (\{p_s, p_e\}, \{t_{emt}\}, \{(p_s, t_{emt}), (t_{emt}, p_e)\}, \{\phi\}, \{\phi\}, \{\phi\})$$

There is often a need to use an activity that does nothing and the empty activity is used for this purpose.

Journal of Theoretical and Applied Information Technology
20th February 2013. Vol. 48 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.2 Modeling structured activities

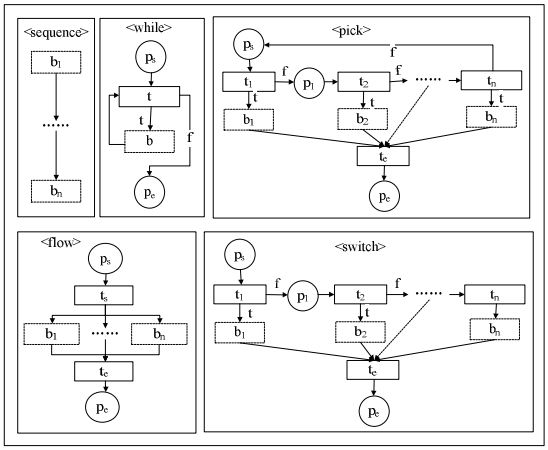

Figure 2: BPN Model Of Structured Activities

(1) sequence

$$BPN_{seq} = \{b_i \mid 1 \le i \le n\}_{seq}$$

where:

- $b_i = (p_{si}, p_{ei}, P_i, T_i, F_i, W_i, V_i, OP_i)$, $1 \le i \le n$
- $p_s = p_{s1}, p_e = p_{en}$
- $P = (\bigcup_{i=1}^{n} P_i \cup P') \setminus \{p_{s(i+1)}, p_{ei} \mid 1 \le i < n\}$

  $P' = \{p_i \mid \dot{p}_i = \dot{p}_{ei}, 1 \le i < n\}$
- $T = \bigcup_{i=1}^{n} T_i$
- $F = (\bigcup_{i=1}^{n} F_i \cup F') \setminus \{(\dot{p}_{ei}, p_{ei}), (p_{s(i+1)}^{\dot{}}, p_{s(i+1)}) \mid 1 \le i < n\}$

  $F' = \{(\dot{p}_{ei}, p_i), (p_i, p_{s(i+1)}^{\dot{}}) \mid 1 \le i < n, p_i \in P'\}$
- $W = (\bigcup_{i=1}^{n} W_i \cup W') \setminus \{w(\dot{p}_{ei}, p_{ei}), w(p_{s(i+1)}, p_{s(i+1)}^{\dot{}}) \mid 1 \le i < n\}$

  $W' = \{w(\dot{p}_{ei}, p_i) \mid w(\dot{p}_{ei}, p_i) = w(\dot{p}_{ei}, p_{ei}) \wedge 1 \le i < n \wedge p_i \in P'\} \cup \{w(p_i, p_{s(i+1)}^{\dot{}}) \mid w(p_i, p_{s(i+1)}^{\dot{}}) = w(p_{s(i+1)}, p_{s(i+1)}^{\dot{}}) \wedge 1 \le i < n \wedge p_i \in P'\}$
- $V = \bigcup_{i=1}^{n} V_i$
- $OP = \bigcup_{i=1}^{n} OP_i$

A sequence activity is used to connect different activity blocks that are performed sequentially. The set of variables in the ending place of b_i is the same with the set of variables in the starting place of b_{i+1}, otherwise the sequence is invalid.

(2) switch

$$BPN_{swc} = \{(c_i, b_i) \mid 1 \le i \le n\}_{swc}$$

where:

- $b_i = (p_{si}, p_{ei}, P_i, T_i, F_i, W_i, V_i, OP_i)$, $1 \le i \le n$
- $P = \bigcup_{i=1}^{n} P_i \cup \bigcup_{i=1}^{n-1} \{p_i\} \cup \{p_s, p_e\}$
- $T = \bigcup_{i=1}^{n} T_i \cup \bigcup_{i=1}^{n} \{t_i\} \cup \{t_e\}$
- $F = \bigcup_{i=1}^{n} F_i \cup \{(p_s, t_1), (t_e, p_e)\} \cup \bigcup_{i=1}^{n-1} \{(t_i, p_i), (p_i, t_{i+1})\} \cup \bigcup_{i=1}^{n} \{(t_i, p_{si}), (p_{ei}, t_e)\}$
- $W = \bigcup_{i=1}^{n} W_i \cup \bigcup_{i=1}^{n-1} \{W(t_i, p_i) = f\} \cup \bigcup_{i=1}^{n-1} \{W(t_i, p_{si}) = t\}$
- $V = \bigcup_{i=1}^{n} V_i$
- $OP = \bigcup_{i=1}^{n} OP_i \cup \bigcup_{i=1}^{n} \{c_i\}$, $c_n = \phi$

The switch activity includes an ordered list of one or more conditional transitions. These transitions are considered in the order of appearance. If the condition c_1 holds true, the activity block b_1 is performed. If no any conditions hold true in $\{b_i \mid 1 \le i \le n-1\}$, b_n is performed.

(3) while

$$BPN_{whl} = \{(c, b)_k\}_{swc}$$

where:

- $b = (p_s', p_e', P', T', F', W', V', OP')$
- $P = P' \cup \{p_s, p_e\}$
- $T = T' \cup \{t\}$, $t.op = c$
- $F = F' \cup \{(p_s, t), (t, p_e), (t, p_s'), (p_e', t)\}$
- $W = W' \cup \{W(t, p_s') = t, W(t, p_e) = f\}$
- $V = V'$
- $OP = OP' \cup \{c\}$
- k is the number of cycles

The while activity iterates performance of a specified activity block b until the given condition c no longer holds true.

(4) flow

$$BPN_{flw} = \{b_i \mid 1 \le i \le n\}_{flw}$$

where:

- $b_i = (p_{si}, p_{ei}, P_i, T_i, F_i, W_i, V_i, OP_i)$, $1 \le i \le n$

Journal of Theoretical and Applied Information Technology
20th February 2013. Vol. 48 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- $P = \bigcup_{i=1}^{n} P_i \cup \{p_s, p_e\}$
- $T = \bigcup_{i=1}^{n} T_i \cup \{t_s, t_e\}$
- $F = \bigcup_{i=1}^{n} F_i \cup \{(p_s, t_s), (t_e, p_e)\} \cup \bigcup_{i=1}^{n} \{(t_s, p_{si}), (p_{ei}, t_e)\}$
- $W = \bigcup_{i=1}^{n} W_i$
- $V = \bigcup_{i=1}^{n} V_i$
- $OP = \bigcup_{i=1}^{n} OP_i$

The flow activity includes concurrent performance of a set of activity blocks $\{b_i \mid 1 \le i \le n\}$. It finishes when all activity blocks terminated.

(5) pick

$$BPN_{pck} = \{(e_i, b_i) \mid 1 \le i \le n\}_{pck}$$

where:

- $b_i = (p_{si}, p_{ei}, P_i, T_i, F_i, W_i, V_i, OP_i)$, $1 \le i \le n$
- $P = \bigcup_{i=1}^{n} P_i \cup \{p_s, p_e\} \cup \bigcup_{i=1}^{n-1} \{p_i\}$
- $T = \bigcup_{i=1}^{n} T_i \cup \{t_e\} \cup \bigcup_{i=1}^{n} \{t_i\}$, $t_i.op = e_i$
- $F = \bigcup_{i=1}^{n} F_i \cup \{(p_s, t_1), (t_e, p_e), (t_n, p_s)\} \cup \bigcup_{i=1}^{n-1} \{(t_i, p_i), (p_i, t_{i+1})\} \cup \bigcup_{i=1}^{n} \{(t_i, p_{si}), (p_{ei}, t_e)\}$
- $W = \bigcup_{i=1}^{n} W_i \cup \bigcup_{i=1}^{n-1} \{W(t_i, p_i) = f\} \cup \bigcup_{i=1}^{n} \{W(t_i, p_{si}) = t\} \cup \{W(t_n, p_s) = f\}$
- $V = \bigcup_{i=1}^{n} V_i$
- $OP = \bigcup_{i=1}^{n} OP_i \cup \bigcup_{i=1}^{n} \{e_i\}$

The pick activity is a set of branches of the activity blocks, and one of the branches will be selected based on the occurrence of the events associated with it. Note that after the pick activity has accepted an activity for handling, the other activities are no longer accepted by that pick. The last activity b_n is triggered by a time-out event e_n.

3.3 Modeling the travel planning process

Travel planning process is shown in Figure 3. When a customer needs to plan his travel, he sends a request to travel planning process by inputting his travel information. The process receives his information and sends use customer ID to retrieve the customer class such as common user, vip user. Finally, the process retrieves ticket prices of American Airline and Air China according to the class ID and returns a cheaper offer to the customer.

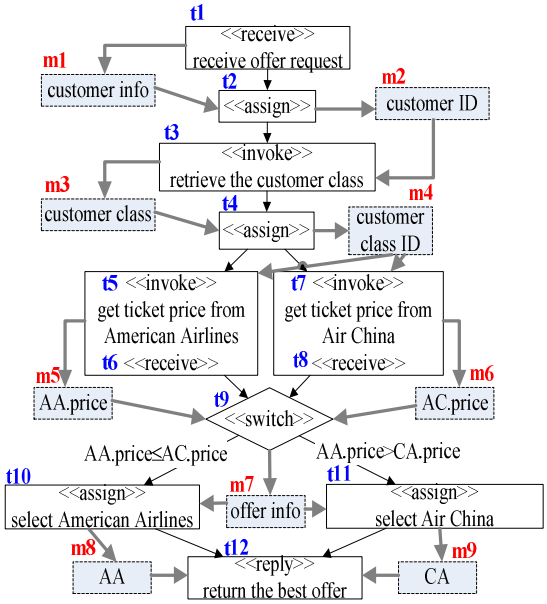

Figure 3: Travel Planning Process

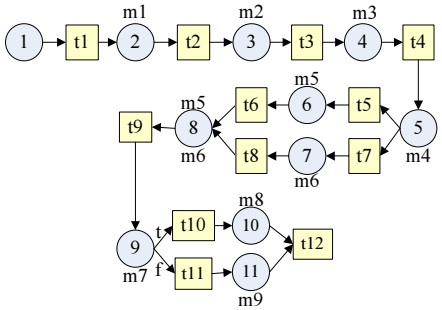

Figure 4: BPN Model For Travel Planning Process

The complete BPN model for the travel planning process is described as follows (also reference to Figure 4):

- $BPN_{t1} = (\{p_1, p_2\}, \{t1\}, \{(p_1, t1), (t1, p_2)\}, \{\phi\}, \{m1\}, \{op_1\})$
- $BPN_{t2} = (\{p_2, p_3\}, \{t2\}, \{(p_2, t2), (t2, p_3)\}, \{\phi\}, \{m1, m2\}, \{copy(m1, m2)\})$
- $BPN_{t3} = (\{p_3, p_4\}, \{t3\}, \{(p_3, t3), (t3, p_4)\}, \{\phi\}, \{m2, m3\}, \{op_2\})$
- $BPN_{t4} = (\{p_4, p_5\}, \{t4\}, \{(p_4, t4), (t4, p_5)\}, \{\phi\}, \{m3, m4\}, \{copy(m3, m4)\})$
- $BPN_{t5} = (\{p_5, p_6\}, \{t5\}, \{(p_5, t5), (t5, p_6)\}, \{\phi\},$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\{m4\},\{op_3\})$

- $BPN_{t5} = (\{p_5, p_6\},\{t5\},\{(p_5,t5),(t5,p_6)\},\{\phi\},$ $\{m4\},\{op_3\})$
- $BPN_{t7} = (\{p_5, p_7\},\{t7\},\{(p_5,t7),(t7,p_7)\},\{\phi\},$ $\{m4\},\{op_5\})$
- $BPN_{t8} = (\{p_7, p_8\},\{t8\},\{(p_7,t8),(t8,p_8)\},\{\phi\},$ $\{m6\},\{op_6\})$
- $BPN_{t9} = \{(AA.price \geq AC.price, BPN_{t10}),$ $(AA.price < AC.price, BPN_{t11})\}_{swc}$

  with
  - $BPN_{t10} = (\{p_9, p_{11}\},\{t10\},\{(p_9,t10),$ $(t10,p_{11})\},\{\phi\},\{m7,m8\},$ $\{copy(m7,m8)\})$
  - $BPN_{t11} = (\{p_{10}, p_{11}\},\{t11\},\{(p_{10},t11),$ $(t11,p_{11})\},\{\phi\},\{m7,m9\},$ $\{copy(m7,m9)\})$
  - $P = \{p_8, p_9, p_{10}, p_{11}\}$
  - $T = \{t9,t10,t11\}$
  - $F = \{(p_8,t9),(t9,p_9),(t9,p_{10}),(p_9,t10),$ $(p_{10},t11),(t10,p_{11}),(t11,p_{11})\}$
  - $W = \{W(t9,p_9) = t, W(t9,p_{10}) = f\}$
  - $V = \{m7,m8,m9\}$
  - $OP = \{copy(m7,m8), copy(m7,m9),$ $AA.price \leq AC.price,$ $AA.price > AC.price\}$
- $BPN_{t12} = (\{p_{11}, p_{12}, p_{13}\},\{t12\},\{(p_{11},t12),(p_{12},$ $t12),(t12,p_{13})\},\{\phi\},\{m8,m9\},\{op_7\})$

4. DIAGNOSIS METHOD

A BPEL process can run down for many reasons. For example, messages mismatch the interface, data format is wrong; Internet is down, and so on. The diagnostic aim is to quickly and exactly find out the faulty activities and analyzes causes by the thrown exception.

For facilitating diagnosis, the BPEL diagnosis process has to be extended for the following tasks:

- record the executed activities in order;
- record the input and output SOAP messages;
- record the thrown exception.

4.1 Diagnosis Model

The BPN model describes the BPEL activities using transitions, variables, places and operations. The BPN-based diagnosis model presents the dependency relations between activities.

Definition 4 a BPN-based diagnosis model for BPEL process is *BDM = (BPN, OBS, D)*, where:

- $BPN = (p_s, p_e, P, T, F, W, V, OP)$
- $OBS = (T', V', OP')$, a set of observations
- $D = \{EQ, EQT, EQV, IN, CO\}$ with
  - *EQ*: an operation in *BPN* equates to the one in *OBS*
  - *EQT*: the data type of a variable in *BPN* equates to another one in *OBS*
  - *EQV*: the value of a variable in *BPN* equates to another one in *OBS*
  - *IN*: the output parameters are produced by invoking another BPEL process
  - *CO*: the computing result of an express or condition

The details of BPN-based diagnosis model are described as follows:

(1) basic activities diagnosis

- receive
  $D(t_{rec}) = \{EQ(op_{rec}, op'_{rec}), EQT(v,v'),$ $EQV(v,v')\}$
- invoke
  $D(t_{inv}) = \{EQ(op_{inv}, op'_{inv}), EQT(v_i, v'_i),$ $EQT(v_o, v'_o), IN(op'_{inv}.service)\}$
- reply
  $D(t_{rpl}) = \{EQ(op_{rpl}, op'_{rpl}), EQT(v,v'),$ $EQV(v,v')\}$
- assign
  $D(t_{asn}) = \{EQT(v_i, v'_i), EQT(v_o, v'_o), EQV(v'_i, v'_o)$ $\vee EQV(CO(v'_i), v'_o)\}$
- wait
  $D(t_{wat}) = \{EQV(CO(c), CO(c')),$ $EQV(c', W'(t'_{wat}, t'_{wat}.op'_{wat}))\}$
- empty
  $D(t_{emt}) = \{EQ(t'_{emt}.op, \phi)\}$

(2) structured activities diagnosis

- sequence
  $D = \bigcap_{i=1}^{n} D_i$
- switch
  $D = \bigcap_{i=1}^{m} \{EQV(CO(c_i), f))\} \cap$ $((\{EQV(CO(c_{m+1}), t)\} \cap D_{m+1}) \cup$ $(EQV(m+1,n) \cap D_n))$
- while
  $D = (EQV(CO(c), t) \cap D_b \cap D) \cup$ $(EQV(CO(c), f))$
- flow:

Journal of Theoretical and Applied Information Technology
20th February 2013. Vol. 48 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$D = \bigcup_{i=1}^{n} D_i$$

- pick:

$$D = \bigcap_{i=1}^{m} \{EVQ(CO(e_i), false)\} \cap (((\{EQV(CO(e_{m+1}), t)\} \cap D_{m+1}) \cup (\{EQV(CO(e_n), f)\} \cap D))$$

4.2 Diagnosis for BPEL Process

When an exception occurs during the execution of BPEL process, we firstly diagnose the exception activity. If the exception activity isn't a fault, we consider the activity with the maximum diagnosis probability as the next activity to be diagnosed. When we find the faulty activity, we end the diagnosis process. The diagnosis probability for activity *t* is computed by

$$DP(t, tf) = \frac{P(\bar{t} \cdot T \cdot t\tilde{f})}{P(t\tilde{f})} = \frac{n(\bar{t} \cdot T \cdot t\tilde{f})}{n(t\tilde{f})} \quad (1)$$

where $\bar{t}$ denotes the activity *t* is fault, $t\tilde{f}$ denotes the activity *tf* is the exception, *T* denotes a set of normal activities and *N* denotes the total count of faulty execution. According to the historical data, we can compute $P(\bar{t} \cdot T \cdot t\tilde{f})$ and $P(t\tilde{f})$. Here $n(\bar{t} \cdot T \cdot t\tilde{f})$ is the number of the executions in which the activity *t* is the fault, *T* is the set of normal activities and *tf* is the exception. $n(t\tilde{f})$ is the number of the executions that the activity *tf* is the exception. In Equation (1), we need to consider how to deal with the branch and loop structure of the process. For example, an activity included in a while activity could be executed several times. We don't consider how many times the activity executed in an execution and only record it once.

Algorithm 1

Algorithm 1: *BPNDiag*(*BPN*,*OBS*,*DS*)
Input: process model *BPN*, Observation *OBS*
Output: diagnosis solution *DS*

```
01: T = null; DS = null; t = OBS.tf;
02: while (T != OBS.T)
03:   if D(t) = fault
04:     DS = DS ∪ {t};
05:     break;
06:   else T = T ∪ {t};
07:   end if
08:   // the activity with the maximum diagnosis probability
09:   t = maxT(DP(OBS.T-T, OBS.tf));
10: end while
11: if DS = null, DS = {OBS.input};
12: return DS;
```

5. EXPERIMENTS

To evaluate the diagnostic efficiency of our method for fault diagnosis, we set up a simulation environment in Matlab. Our simulation environment incorporates three parts. Part one is to build a BPEL process according to the given numbers of activities and structured nodes. Part two is to generate the execution traces of this process according to the given faulty probability. Part three is our diagnosis algorithm. For each generated process, we randomly select an activity as the exception for each process, and another activity as the fault. And we define three fault types: type fault, value fault and port fault. Then we randomly select a fault type and assign it to the faulty activity. Each faulty activity and exception activity is devised to complete successfully at a probability of 0.2, and other activities are devised to always complete successfully. According to the given faulty probability, we generate 100 execution traces for each process. Finally, we generate 1000 processes with different numbers of activities and 100 groups of test data for each process to evaluate our method.

In our experiments, we use the accuracy of diagnosis and diagnostic time as our evaluation metrics and compare our method with a model-based diagnosis method. Following is the description of labels we use to denote two methods:

- ard: This is a model-based diagnosis method Ardissono proposed in [12].
- bpn: This is our diagnosis method.

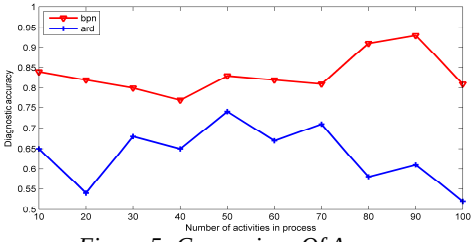

Figure 5: Comparison Of Accuracy

The results of comparison for diagnostic accuracy are shown in Figure 5. We can see that the accuracy of our method is much higher than Ardissono's method. The diagnostic accuracy of Ardissono's method is 52%-74%, while the diagnostic accuracy of our method is 77% to 93%. That is because our method considers more fault types in our diagnosis model.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

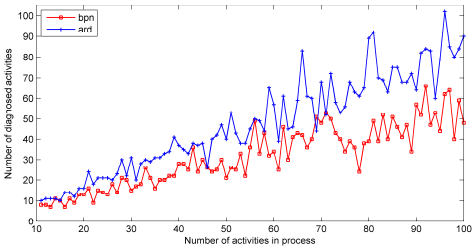

Figure 6: Comparison Of Diagnosed Activities' Numbers

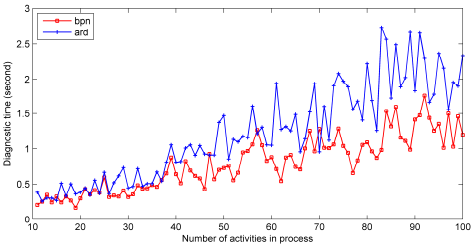

Figure 7: Comparison Of Diagnosed Time

From Figure 6, we can see that our method diagnoses fewer activities than Ardissono's. Hence, our method costs less diagnostic time than Ardissono's method and the results of comparison is shown in Figure 7. This is mainly because our method is not to diagnose one by one according to the observation sequence, but to select the diagnosed activities according to the diagnosis probability. Hence, our method reduces the diagnostic time through decrease of the number of diagnosed activities.

The above experiments show that our method is more effective than Ardissono's method in diagnosis of BPEL process.

6. RELATED WORK

In order to enhance fault management in composite web services with the ability of reasoning on global failures of the overall service, Ardissono et al. [13] proposed a framework which integrated diagnosis services in the architecture of a composite service. The framework includes a global diagnosis service associated with the composite web services process and some local diagnosis services. Each local diagnosis service was responsible for a web service. Local diagnosis services cooperate with global diagnosis service. And a global diagnosis service exchanges messages with them, without relying on any internal structure information of local diagnosis services. This method recursively partition web services into aggregations of sub-services, hide the details of the aggregation to higher-level services so that it could raise diagnostic efficiency and ensure the privacy of service. However, when web services claim too coarsely, almost all web services could be faults, diagnosis services needs to check each service, and these cause diagnostic efficiency depression. Li et al. proposed [8] a diagnostic method for BPEL processes, where processes are modeled using colored Petri nets. According to color propagation functions, the diagnosis service checks back from where system throws the exception until arriving at a final consistency. Yan et al. [6] applied synchronized automaton to model the BPEL process of web services composition. After an exception is thrown, the diagnostic service calculates the process execution track by comparing model with observation. The diagnosis service finds out fault according to given the rules. Mayer et al. [14] proposed a method to identify the fault of web services composition by observations obtained from partial executions and re-executions of a process are exploited. Lakshmi and Mohanty [15] proposed a method based on stochastic automata model for web service fault monitoring and diagnosis.

7. CONCLUSION

In this paper, we propose a BPN model-based diagnosis method for the BPEL process. The dependency relations between activities are completely modeled by BPN. Our diagnosis method localizes the faulty activity and explains faulty reason according to five dependency rules. Moreover, we use the diagnosis probability to fix the diagnostic sequence for reducing the diagnostic time. Finally, we conduct the experiments to compare the accuracy and efficiency of our method with a MBD method. Experimental results show that our method is more effective than the MBD method.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (61175056), the Dalian Maritime University Backbone Youth Foundation (NO.2011QN033, No.2009JC29), and IT Industry Development of Jilin Province.

REFERENCES:

[1] L. Ardissono, L. Console, A. Goy, et al., "Enhancing Web Services with Diagnostic Capabilities", Proceedings of the Third European Conference on Web Services, *IEEE Computer Society*, 2005, pp. 182-191.

[2] G. Friedrich, M. Fugini, E. Mussi, et al., "Exception Handling for Repair in Service-Based Processes", *IEEE Transactions on Software Engineering*, Vol. 36, No. 2, 2010, pp. 198-215.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[3] X. Han, Z. Shi, W. Niu, et al., "Similarity-Based Bayesian Learning from Semi-structured Log Files for Fault Diagnosis of Web Services", *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2010, pp. 589-596.

[4] O. Kopp, F. Leymann, D. Wutke, "Fault Handling in the Web Service Stack", *Service-Oriented Computing*, Vol. 6470, 2010, pp. 303-317.

[5] Z. Zhu, W. Dou, "QoS-Based Probabilistic Fault-Diagnosis Method for Exception Handling", New Horizons in Web-Based Learning: Icwl 2010 Workshops, Springer-Verlag Berlin, 2011, pp. 227-236.

[6] Y. Yan, P. Dague, Y. Pencole, et al., "A Model-based Approach for Diagnosing Faults in Web Service Processes", *The International Journal of Web Services Research (JWSR)*, Vol. 6, No. 1, 2009, pp. 87-110.

[7] Y. Li, T. Melliti, P. Dague, "Modeling BPEL Web services For Diagnosis: towards self-healing Web services", Proceedings of 3rd International Conference on Web Information Systems and Technologies (WEBIST'07), 2006, pp. 1-62.

[8] Y. Li, L. Ye, P. Dague, et al., "A Decentralized Model-Based Diagnosis for BPEL Services", Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society, 2009, pp. 609-616.

[9] Y. Dai, L. Yang, B. Zhang, et al., "Exception diagnosis for composite service based on error propagation degree", 2011 IEEE International Conference on Services Computing, SCC 2011, IEEE Computer Society, 2011, pp. 160-167.

[10] M. Tony Andrews, I. Francisco Curbera, S. S. Hitesh Dholakia, et al., "Business Process Execution Language for Web Services", 2003, pp. 1-136.

[11] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, 1989, pp. 541-580.

[12] L. Ardissono, S. Bocconi, L. Console, et al., "Enhancing Web Service Composition by Means of Diagnosis", Business Process Management Workshops, 2008, pp. 468-479.

[13] L. Ardissono, L. Console, A. Goy, et al., "Cooperative Model-Based Diagnosis of Web Services", Proceedings of 16th International Workshop on Principles of Diagnosis 2005.

[14] W. Mayer, G. Friedrich, M. Stumptner, "Diagnosis of Service Failures by Trace Analysis with Partial Knowledge", *Service-Oriented Computing*, Vol. 6470, 2010, pp. 334-349.

[15] H. N. Lakshmi, H. Mohanty, "Automata for Web Services Fault Monitoring and Diagnosis", *International Journal of Computer & Communication Technology (IJCCT)* Vol. 3, No. 2, 2011, pp. 13-18.