INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 12, NO. 1, MARCH 2007, 37-44

Verifying Dynamic Workflow Change based on Executable Path

Jian CAO, Haiyan ZHAO, Jie WANG, Shensheng ZHANG, Minglu LI

Abstract- **As a technology that can improve the efficiency of the business process, workflow is drawing more and more attentions of researchers and software product vendors. But there are still many problems waiting to be solved for workflow. One of these problems is the poor adaptability of workflow. This paper intends to enhance the adaptability from the view of supporting dynamic change on process model for workflow instance. In the paper, a workflow model that is fit for dynamic change is defined firstly and the verification rules are set up for this model. Specifically, our method is based on the concept of executable path of workflow. Based on the executable path, the concepts and algorithms of valid process and complete sub-process are introduced to solve the verification problem brought by workflow instance dynamic change.**

***Index Terms*—Adaptive Workflow, Executable Path, Valid Process, Complete Sub-Process**

1. INTRODUCTION

Today's dynamic business environment is driving organization, which competes globally, to focus on lowering prices and customizing products and services. A key success factor for effective competing one is the management of core business processes, which deliver value to their customers, suppliers and internal staffs. Thus by focusing on automating, optimizing, and continuously improving the core business processes, organizations can make commitments to those customers, employees, partners, and suppliers establishing a solid competitive advantage [6]. Workflow technology that originated from office automation catches this trend, so in recent years both researchers and software vendors attach great importance to it [7][17].

Although workflow has been applying widely, it has not obtained anticipated success and there is a gap between the products available in market and requirements [9]. Why commercial WfMS can't meet the market needs? Hammer and Champy pointed out that modern enterprise had three characters: (1) user dominating; (2) intensive competition; (3) continuous change [3]. From the view of workflow technology, user dominating means a system must support the ad-hoc process, continuous change means dynamic process model. In the competition world of "only change is unchangeable", the ability to deal with the ad-hoc process and dynamic process model is necessary for a successful enterprise. Unfortunately current workflow can't provide full supports to the enterprise in this aspect.

More and more researchers begin to realize the problems of the current WfMSs and they put out the concept of adaptive workflow to solve this problem. In recent years, some international conferences were held to discuss the adaptability of the workflow [14]. At the same time, the papers related to the adaptive workflow are increasing greatly. Adaptive workflow has been becoming a research focus now.

There are two methods to improve the adaptability of a WfMS. One is called a-priori flexibility that means to define all possible cases during the modeling phase. The other is called a-posteriori flexibility and in this method process model change is allowed when the instance of workflow is running [9]. Since it is very difficulty to enumerate all situations in advance if not impossible, a-posteriori method is more realistic than a-priori one.

An important issue corresponding to dynamic change is how to assure the correctness of the modified process model of workflow instance, i.e., the process of workflow instance can be completed and each meaningful activity for the remaining process has a way to be executed. When a dynamic change happens, such as to add or remove an activity for an already running workflow instance's process model, the fact that some activities have been executed or are executing should be considered carefully. During the executing process, the workflow instance's process model is also being change at the same time, for example the conditional branch with certain execution probability becomes deterministic when some value has been set. It makes the problem of verifying dynamic change different from that of process model verification in modeling phase. In many systems special change operations are defined and if only these operations are applied to modify the process model, it's correctness can be maintained [11]. The shortcoming of this approach is it often limits the possible changes user can make. The method presented in this paper allows a user to change the process model of workflow instance freely, which means he can take all operations applied in creating a new workflow model. Then algorithms are applied to verify the correctness of the changed process model.

The paper is organized as follows. Section 1 is an introduction. In section 2, the workflow model and its

This work was supported by China NSF under Grant No. 60503041 and the National High-Tech Research and Development Plan of China under Grant No. 2006AA04Z152 This paper is extended from "Verification of Dynamic Process Model Change to Support the Adaptive Workflow" published at *IEEE International Conference on Services Computing*, 15-18 Sep. 2004, Shanghai, China.

Jian CAO, Shensheng ZHANG, Minglu LI are with Dept. of Computer Science and Engineering, Shanghai Jiaotong University, 800, Dongchuan Road, Minghang, Shanghai, China (e-mail: cao-jian@cs.sjtu.edu.cn, sszhang@sjtu.edu.cn and li-ml@cs.sjtu.edu.cn); Haiyan ZHAO is with School of Computer Engineering, University of Shanghai for Science and Technology, 516, Jungong Road, Shanghai, (email: zhaohaiyan92@gmail.com). Jie WANG is with Civil and Environment Engineering, Stanford University, CA 94305, USA(e-mail: jiewang@stanford.edu)

correctness rules are defined firstly as the basis of the paper. Section 3 introduces the concept of executable path, which can be applied to verify a process model. In section 4 and 5, the algorithms and concepts are discussed to deal with the verification problem of dynamic process model change. Section 6 is the related work. Section 7 briefly summarizes the whole paper and points out several directions for future work.

2. A WORKFLOW MODEL SUPPORTING DYNAMIC CHANGE

Most graphic workflow model such as ADEPT defines the logic relationships among activities through their input characters (AND-join, OR-join) and output characters (AND-split, OR-split) [11]. It makes logic relationships among activities be closely coupled with activities themselves. Since model change usually only means to adjust the logic relationships among activities, the coupling relationships among activities and their logic relationships lead to this kind of change more difficult. In the workflow model defined below, activities and their logic relationships are decoupled and activities must be connected with each other through some logic nodes.

Definition 1: A workflow process model can be defined as WM=(E, A, C, D, R, *TC*, *RE*, RC, CF, DF), where E is the event set, A is the activity set, C is the logic node set, D is the data object set, R is the condition set. $TC \in C \rightarrow \{<AND, ALL>, <AND, XOR>, <OR, ALL>\}$ is a function that maps the logic node c to its type, i.e., $\forall c$, $TC(c) \in \{<AND, ALL>, <AND, XOR>, <OR, ALL>\}$, $c \in C$. In the type expression $<L_1, L_2>$, the first element L_1 represents the input logic of the logic node and the second element L_2 is the output logic of the logic node. RC is the corresponding relationships between condition and logic node whose type is <AND, XOR>. *RE* is a function to map the satisfying state of a condition to an event. $CF=\{E\times A\times C\}\vee\{E\times C\times A\}\vee\{E\times C\times C\}$ is the control flow set and e in $cf \in CF$ is called control flow driven event. $DF=\{E\times A\times A\times D\}$ is the data flow and e in $df \in DF$ is called dataflow driven event.

Fig.1 gives out an example of workflow model for part design. The figure doesn't show the event information, which is defined as properties of control flows and data flows.

According to the definition 1, we can find this model is an event driven model and it is quite straightforward to transform it into an ECA rule based workflow model [4]. An activity can be triggered if it's trigger events happen and be detected by workflow engine [4]. This model separates the logic relationships from the activities themselves and provides conveniences to the process change.

If we regard each element in set A and C as a vertex of a graph and each link (control flow) as a piece of edge, the workflow model is a directed graph, which is called directed process graph. Suppose the vertex set of the directed graph is NS, then $NS=A\cup C$ and the edge set is $AS=NS\times NS$.

There are two special logic activities, i.e., A_s and A_e. A_s is called logic start activity and A_e is called logic end activity.

Definition 2: A workflow process is logic correct if and only if:

For any activity a, there is at least one executing sequence of activities including a and starting from A_s, which can trigger activity a.

For any activity a, there is at least one executing sequence of activities including a, in which process can be ended at activity A_e.

This definition means on the one end, each activity should be reachable and on the other hand, after it is executed, the process can be ended.

Definition 3: For any two activities a_1 and a_2, if there is a data flow $\{e\times a_1\times a_2\times D\}$, then a_1 should have a chance in which it happens before a_2. If this condition can be met, then we call this workflow process is correct in data flow definition.

Definition 4: A workflow process is correct if and only if the workflow process is correct both in logic and data flow definitions.

The verification of the correctness of the data flow is relatively simple. If there is a data flow from activity a_1 to a_2, then we only need to check if there is a case in which a_1 can be executed before a_2. Therefore the verification issue of the data flow will not be discussed in this paper. We will also not differentiate the correctness of workflow process and logic correctness in following sections.

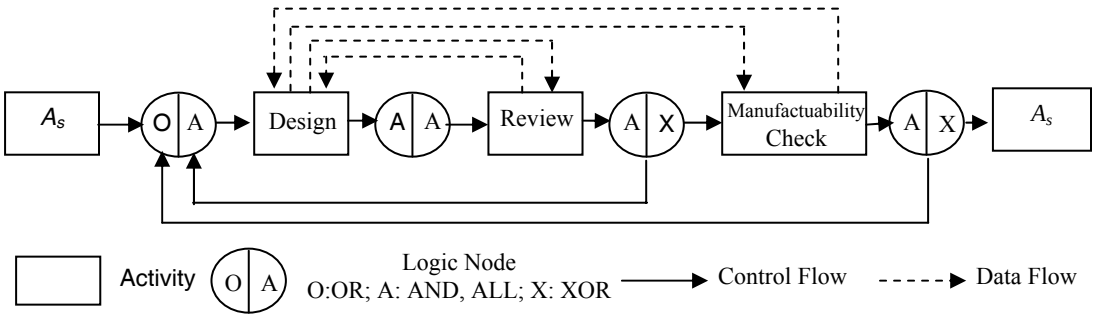

Fig. 1. An Example of Workflow Model

3. THE EXECUTABLE PATH AND ITS SEARCH METHOD

Definition 5: To all $n \in NS$, $\bullet n=\{m| (m, n) \in AS\}$ is the input vertex set of vertex n. $n \bullet=\{m| (n, m) \in AS\}$ represents the output set of the vertex n.

Rule 1: For any logic node with type <AND, XOR>, only one condition can be satisfied for every possible value assignments of the variables defined in the corresponding conditions.

This rule is used to restrict that the output of this kind of logic node is mutually exclusive.

Rule 2: In any workflow process model, there are only one A_s and one A_e.

Rule 3: $|a \bullet| \geq 1$ and $| \bullet a | \geq 1$, $a \in A$

$|c \bullet| \geq 1$and $| \bullet c| \geq 1$, $c \in C$

$(\bullet a) \in C$, $(a \bullet) \in C$

$|A_s \bullet| =1$ and $|\bullet A_s| =0$; $|A_e \bullet| =0$ and $| \bullet A_e| =1$.

This rule defines the legal structure of a workflow process model.

Definition 6: The executable path p is a vertex sequence $\langle n_i, n_{i+1}, \ldots, n_j\rangle$ that is connected by end-to-end edges. In this sequence, we have $\langle n_k, n_{k+1}\rangle \in AS$, $i \leq k \leq j-1$.

Suppose NS(p) is the vertex set of the path p.

Definition 7: In an executable path $p=\langle n_i, n_{i+1},..., n_j\rangle$, if $n_i=n_j$, then it is called a loop.

Definition 8: A complete path is an executable path starting from the A_s and ending at the A_e and also it must not include any loop.

Definition 9: To an executable path $p=\langle n_i, n_{i+1},..., n_j\rangle$, the distance between two vertexes is defined as the number of vertex between these two points in the path plus 1.

The following algorithm is used to search all the executable paths in a process model.

Algorithm 1:

Suppose the value domain of variable v_i is the output set $n_i\bullet$ of the corresponding vertex n_i, then we can get a vector$<v_1, v_2, \ldots>$. Denote the value domain of this vector as V. Suppose CP, RS are two empty sets.

To each value of V, suppose it is $<v_{c1}, v_{c2},, \ldots, v_{ct},\ldots>$, the search process is as follows:

<1>Create an empty path p;

<2>Add the A_s to the tail of the path p;

<3>Obtain the last point of the path p, suppose it is n_t;

<4>From the $<v_{c1,} v_{c2},\ldots,v_{ct}, \ldots>$ to obtain the corresponding value v_{ct} of n_t:

If v_{ct} has been existed in p, then record the v_{ct} to the tail of the p as a loop, and add this loop to the RS, search process is ended.

If v_{ct} is A_e, add the A_e to the tail of the p and then we obtain a complete path, add the p to the CP, search process is ended;

Otherwise, add the v_{ct} to the tail of the p, return back to <3>

After the search process, CP and RS can be obtained.

Currently, the verification of the workflow process model often adopts method of the Petri Net [15-17]. In order to apply this method, a workflow process model should be turned into Petri Net. The executable path provides another general way to check the correctness of a process model and also offers some advantages for verification of dynamic process modification, which will be discussed in section 4 and 5.

Some errors can be detected directly by the search algorithm above:

Rule 4: To a variable v_i of the vector $<v_1, v_2, \ldots>$, if the values of other variables are not changed and v_i has been tried any value of its domain, there is no complete path can be found then the corresponding vertex of v_i in process graph will lead to this process can not be ended.

Rule 5: If a vertex is neither in CP nor in RS, then this vertex is not accessible.

Besides the errors detected according to rule 4, improper definition of logic nodes in the process model can also lead to deadlock and cause a process can not be ended. Especially, the semantics of node <AND, XOR> should be considered carefully. Node <AND, XOR> can be divided into two types. The first type is the values of the variables in the conditions defined for the node will not be changed during the execution of the process instance. In the other type, the values of the variables can be changed during the execution of the process instance. To the second one, some parts of process may be repeated and the values of the variables changes so that it make the process executing leave from the loop and the whole process can be ended. For example, in Fig.1, if in the review the engineer does not approve the design, than the process executing returns to the design activity through the node <AND, XOR> and after the second time review the design may be approved so that the process executing continues to the next step. In this situation, the process model is also correct.

The algorithm below is designed to detect the dead lock.

Algorithm 2:

Suppose x_i is corresponding to node n_i whose type is <AND, XOR> and the values of variables in the relating conditions will not be changed during the process executing. The value domain of x_i is $(n_i \bullet)$. Thus all these types of nodes are corresponding to a vector $<x_1, x_2, \ldots>$. The value of the vector will compose the domain X. Suppose y_i is corresponding to the node n_j whose type is also <AND, XOR> and the values of variables in the related conditions can be changed during a process. Suppose the value domain of y_i is $(n_j \bullet)$. Thus all this kind of nodes is corresponding to a vector $<y_1, y_2, \ldots>$. The value of the vector will compose the domain Y. Suppose AN is a FIFO stack, EN is an empty set.

To a value in X:

<1>Take out a point from Y;

<2>Assume A_s is activated, then $A_s \bullet$ can be added to AN and A_s can be added to set EN;

<3>$k=0$;

<4>Take out an element n_i from AN and let k=k+1. Judge whether the activation conditions for n_i can be satisfied:

If conditions can be satisfied then k=0.

If n_i is A_e then the algorithm can be ended

Else: According to the values of <x_1, x_2, …> and <y_1, y_2, …>, add the value of variable corresponding to n_i• to the AN and add n_i to the set EN. Repeat<4>.

else

if k is equal to the element number of AN then go to <5>;

Else: Add the n_i to AN and repeat<4>

<5>if there is other value of Y that has not been tried then select another point in Y and return to <2>

else: end.

After calculating, we have obtained a validating rule as follows:

Rule 6: If the value of X is fixed and the value of Y can be anything, the process can't be ended so that the process has a deadlock.

The computation complexity of some verification problem has been proven in [13]. The initiation problem for a workflow object x in a workflow structure W is to determine whether a sequence of events exists leading to the execution of x. The termination problem is to determine whether a workflow structure can reach a terminal state. A workflow structure is safe if and only if from every reachable state a terminal state can be reached. They proved in the paper that the initialization problem is NP-complete, any algorithm solving the termination problem and determining safeness will require at least an exponential amount of storage space. Since what we are try to verify by applying Rule 4, 5 and 6 fall into these three categories, the computation complexity follows their conclusions.

Verification method provided above should be carried out before a workflow model is running, thus it is a kind of offline verification. Dynamic change, such as adding activity, deleting activity and adjusting the logic relationships among activities, may happen when some instances of a workflow are already running. During dynamic changing, two problems should be tackled. Firstly, change operations will have effect on the running case. Secondly, to assure the change is correct, i.e., the change can't cause the error of the process. It seems that algorithms introduced above can solve the second problem. But verification process is very complex in computation. In many situations, the verification needs not to be carried out for the whole model. Therefore in the following two sections, the concepts of valid process and complete sub-process and their related algorithms are introduced.

4. THE VALID PROCESS AND ITS SEARCH METHOD

For a running workflow instance, when an activity has been executed and it will never be executed any more, to delete it from the process model of this instance is a useless operation. Therefore, we can remove such activities from the process model of this workflow instance so that we can assure that any change we take has effect on this instance. At the same time, reduced process model also simplifies the verifying process of the correctness of the model.

Definition 10: For each running instance of a workflow, the activities that can be executed and their relationships compose a valid process model, in which an activity is executing or can be executed in the future.

Proposition 1: For a change, if a changed valid process model is correct, then the changed whole process model is also correct

Theorem 1 is obvious. This property also shows that after some changes, to verify the correctness of whole process model only needs to verify the valid process model of the running instance.

The algorithm below is designed to find the valid process model for a running instance.

Algorithm 3:

Suppose the set of executing or executed vertexes is EN and p_i=<A_s, n_{i2}, …, n_{ij}, n_{ij+1}, …, A_e>is a complete path.

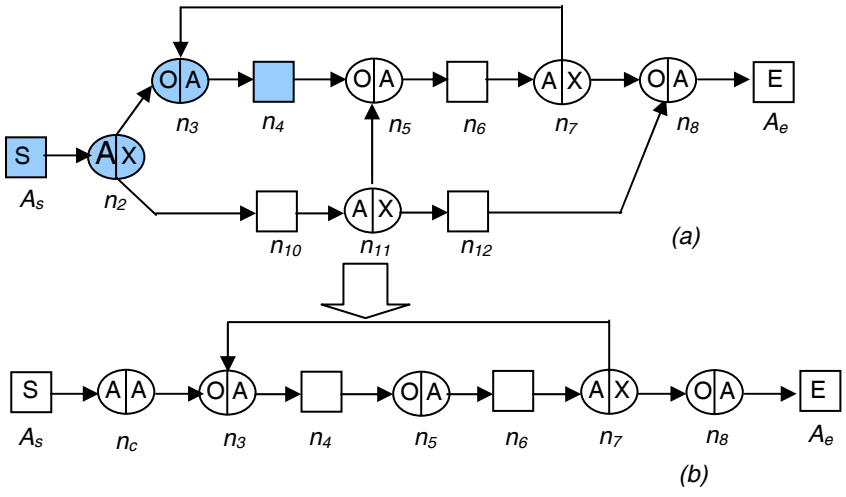

Fig. 2 An Example of the Valid Process Model

<A_s, n_{i2}, …, n_{ij}> is a partial path from A_s to n_{ij}, where $n_{ij}\in$EN and $\{n_{ij+1}, …, A_e\}\cap$EN=ϕ. The nodes of p_i can be divided into two sets P_{ai} and P_{bi}, where $P_{ai}=\{A_s, n_{i2}, …, n_{ij}\}$ and $P_{bi}=\{n_{ij+1}, …, A_e\}$. Suppose there are altogether m pieces of complete paths in CP:

<1>i=1;

<2>To a complete path p_i, assume k=1;

<3>k=k+1;

<4>if k<j then search all the loops to judge whether there is a loop ro_t satisfying $n_{ik}\in$NS(ro_t) and NS(ro_t)$\cap p_{bi}\neq$ϕ:

If this kind of loop exists, then turn to <5>;

Otherwise delete n_{ik} from p_i, turn to <3>;

If k=j and n_{ij} is <AND, XOR> then delete p_i from CP and turn to <5>;

<5>i=i+1, if i>m then turn to the next step otherwise turn to <2>;

<6>Add <AND, AND> logic node n_c, add n_c to each path of CP and its position is next to A_s;

<7>According to the set of complete path and loops, the process flow can be figured out.

Fig. 2 is an example of the valid process model. Fig. 2 (a) is a process model, suppose the nodes with shadow have been executed.

In the process model, there are altogether three pieces of complete executable path, they are:

p_1: A_s, n_2, n_3, n_4, n_5, n_6, n_7, n_8, A_e

p_2: A_s, n_2, n_{10}, n_{11}, n_{12}, n_8, Ae

p_3: A_s, n_2, n_{10}, n_{11}, n_5, n_6, n_7, n_8, A_e

r_{o1}: n_3, n_4, n_5, n_6, n_7, n_3

Suppose the state of n_4 is executing now, then EN=$\{A_s, n_2, n_3, n_4\}$. According to the algorithm, the path and loop are as follows:

p_1: A_s, nc, n_3, n_4, n_5, n_6, n_7, n_8, A_e

r_{o1}: n_3, n_4, n_5, n_8, n_7, n_3

According to p_1 and r_{o1}, the valid process model is like Fig. 2 (b).

5. THE COMPLETE SUB-PROCESS MODEL

In the process model, there is a special group of nodes. In this piece of process, there are two distinct nodes called start node and end node and all the edges inputting to the piece are inputting through the start node and all the output edges to the outside of this process piece will output from the end node. We call this process piece a complete sub-process. Obviously, the whole workflow model is a special complete sub-process.

Definition 11: Suppose N is the node set, there are two nodes n_s, n_e, (n_s •)⊆N, (• n_e)⊆N, any $n\in$N\{n_s, n_e}, (n •)⊂N, (• n)⊂N, then all the nodes of N and all the edges among them compose a complete sub-process.

Proposition 2: For a change happened in a complete sub-process, if this complete sub-process model is correct, then the whole process model is also correct.

This proposition is also obvious. It shows when a change happens, verification only needs to be carried out on the smallest complete sub-process affected by this change.

From the viewpoints of the executable path, the complete sub-process has such characters: for all paths in the sub set SP of the path set CP, there is a sub-sequence starting from n_s and ending with n_e and all the vertexes included in the sub-sequence only appear in SP.

According to this idea, an algorithm is designed to find the smallest complete sub-process.

Algorithm 4:

Suppose the vertex set affected by a change is RN. Search all the paths including elements of RN from the CP and they compose a set SP:

<1>Any $ep_i\in$SP can be divided into three pieces: ep_{i1}=<A_s, n_{i2}, …, n_{im}>, ep_{i2}=<n_{im+1}, n_{im+2}, …, n_{in}>, ep_{i3}=<n_{in+1}, n_{in+2}, …, A_e>, the corresponding sets of vertex in each piece can be represented by NS(ep_{i1}), NS(ep_{i2}), NS(ep_{i3}), where $n_{im+1}\in$RN, $n_{in}\in$RN and NS(ep_{i1})$\cap$RN=ϕ, NS(ep_{i3})$\cap$RN=ϕ;

<2>Obtain the interaction set among all NS(ep_{i1}) and all the paths of SP and the result is PPH. Obtain the interaction set among NS(ep_{i3}) and all the paths of SP and the result is PPT;

<3>Suppose the element of PPH can be denoted as n_t. Calculate the distance from n_t to n_{im+1} for each ep_i, get the

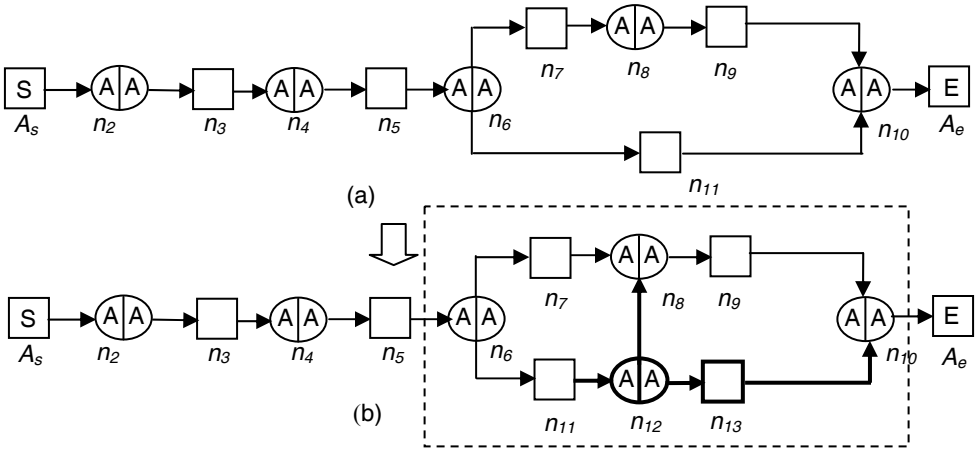

Fig.3 An Example of the Complete Sub-process

sum of these distances for all ep_i and denote it as d_t. Sort the elements from *small* to big according to the value of d_t;

<4>To any element n_s of PPT, calculate the distance between n_s and n_{in} for all ep_i, get their sum for all ep_i and its value is denoted as d_s. Sort all the points of PPT from small to big according to the value d_s;

<5>k=1,j=1;

<6>Take out k th vertex from PPH and denote as n_k, take out the j th vertex from PPT and denote it as n_j;

<7>Take out all the points between n_k and n_j of ep_i of SP and make conjunction for all ep_i. The result is φ;

<8>Check whether all the paths of CP including vertexes of φ is in the SP:

If there is any path not in SP then add the path to SP and return to <1>;

If a vertex of φ is in a loop, then add all the paths including the vertex of this loop to the SP and return to <1>;

else the algorithm is ended. All the paths starting from n_k and ending at n_j of each complete path of SP compose the complete sub process.

Fig.3 is an example of complete sub process.Fig. 3(*a*) is the former process model and after modification the model is changed to Fig. 3(b). The complete executable paths are as follows:

p_1: A_s, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, A_e

p_2: A_s, n_2, n_3, n_4, n_5, n_6, n_{11} , n_{12}, n_{13}, n_{10}, A_e

p_3: A_s,n_2, n_3, n_4, n_5, n_6, n_{11}, n_{12}, n_8, n_9, n_{10}, A_e

The nodes affected by the modification are: RN={n_{12}, n_8, n_{13}}, then SP={p_2, p_3}.

NS(p_{21})={A_s, n_2, n_3, n_4, n_5, n_6, n_{11}}

NS(p_{22})={n_{12}, n_{13}}

NS(p_{23})={n_{10}, A_e}

NS(p_{31})={A_s, n_2, n_3, n_4, n_5, n_6, n_{11}}

NS(p_{32})={n_{12}}, NS(p_{33})={n_8, n_9, n_{10}, A_e}

NS(p_{21})∩NS(p_{31})={A_s, n_2, n_3, n_4, n_5, n_6, n_{11}}

NS(p_{23})∩NS(p_{33})={n_{10}, A_e}

The nodes set from n_{11} to n_{10} {n_{12}, n_{13}, n_8, n_9} interacts with p_1 then:

NS(p_{11})={A_s, n_2, n_3, n_4, n_5, n_6, n_7}

NS(p_{12})={n_8, n_9}, NS(p_{13})={n_{10}, A_e}

NS(p_{11})∩NS(p_{21})∩NS(p_{31})={A_s, n_2, n_3, n_4, n_5, n_6}, NS(p_{13})∩NS(p_{23})∩NS(p_{33})={n_{10}, A_e}

The node set from n_6 to n_{10} is {n_7, n_8, n_9, n_{11}, n_{12}, n_{13}} which can satisfy the conditions, thus the complete sub-process is in the rectangular of the figure 3(b).

6. RELATED WORK

Workflow model verification has long been identified as an important and challenging issue in workflow research. In [13], the hardness of workflow model verification is analyzed. Some works are done based on Petri Net[15-17], set theory[2] or graph reduction techniques [5][10][12]. Tools are also developed for workflow verification [17]. Comparing with them, the algorithms presented in Section 3 is based on executable path search, which can check initialization, termination and safeness problems defined in [13]. The executable path provides the basis for dynamic change verification.

In order to adapt to changes, a-priori and a-posteriori are two ways to enhance the workflow flexibility. A typical example of the first way introduced in [8] is to provide a rule-based mechanism which allows for a given semantic exception to compute the set of workflow instances and the required modifications of these instances to cope with the exceptional situation. However, the exceptions and the respective rules to cope with the exceptions have to be known at workflow modeling time.

The example of the second way is ADEPT system. In order to support workflow change and also maintain the correctness of changed workflow, a formalism which consists of a set of rules, criteria, and methods which support the dynamic modification of workflow instances is defined [11]. In this framework, change operations to the structure of running workflows can be performed by users in a controlled manner.

Workflow evolution, which allows the running workflow instances to migrate from one schema to another schema, can be done both in a-priori and a-posteriori ways [1][18], in which specific modification operations (called workflow evolution primitives) are applied to adapt (or migrate) the workflow instance to a modified workflow schema [1]or to check whether the workflow instance under consideration can be continued to become a complete and correct workflow instance with respect to the new schema [18].

The method presented in the paper is different from those works. Our method allows a user to change the running workflow instance process model freely. Then algorithms will be applied to verify it's correctness.

7. CONCLUSIONS AND FURTHER WORK

An important problem in current workflow system when it is applied into the business is its poor adaptability. The objective of this paper is to provide the user with the ability to change the process model without any restriction. In order to achieve this goal, a formal workflow process model is defined as the basis of the paper. After the process model is changed, two issues should be addressed. The first one is how to judge the effects of a process change to the current running instance and the other one is how to verify the correctness of modified process model.

To solve the first problem, the concept of valid process is introduced as the basis to judge the effect caused by process change. Generally, there will be more than one instances of a workflow running at the same time while staying at different stages. The method provided in the paper can deal with the change on one instance. To solve the dynamical change problem for more than one instance can also be based on the valid process. This is beyond the scope of this paper.

To solve the second problem, the paper introduces the concept of complete sub-process to reduce the computing

complexity of verification. The valid process model can also be applied to simplify the process model so that the computation scale is reduced. The effects of both methods make the verification problem for dynamic process change relatively simpler

ACKNOWLEDGEMENT

The authors would like to thank the comments provided by the anonymous reviewers and editor, which help the authors improve this paper significantly.

REFERENCES

[1] Casati, F., Ceri, S., Pernici, B., Pozzi, G., "Workflow evolution", *Data and Knowledge Engineering*, Vol.24(3), 1998, pp211–238

[2] Dongsoo H., Sundoke L., Minkyu L. , Jaeyong S., "Set-based analysis of structured workflow definition", *International Journal of Cooperative Information Systems*, Vol.14, No.4, 2005, pp503-527

[3] Hammer M., Champy J. , "Reengineering the corporation-a manifesto for business revolution", Harper Business, 1993

[4] Jian, C., Jie, W., Shensheng, Z., Minglu L., "A dynamically reconfigurable system based on workflow and service agents", *Engineering Application of Artificial Intelligence*, Vol.17, No.7, 2004, pp771-782

[5] Lin H, Zhao Z B, Li H C, et al. "A novel graph reduction algorithm to identify structural conflicts", *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS'02)*, Los Alamitos, CA, Sep. 2002, pp289-298

[6] Mentzas, G., Halaris, C., Kavadias, S., "Modeling business processes with workflow systems: an evaluation of alternative approaches", *International Journal of Information Management*, Vol.21, No.1, 2001, pp.123-135

[7] Mohan, C., "Recent trends in workflow management products, standards, and research", *Proceedings of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability*, Istanbul, Turquie, 12-21 August 1997, pp. 396-409

[8] M¨uller, R., Rahm. E., "Rule-based dynamic modification of workflows in a medical domain". Buchmann (Editor): *Proc. of BTW'99*. Informatik Aktuell Berlin, Springer, 1999, pp429–448

[9] Ouksel A. M., et al, "The Need for adaptive workflow and what is currently available on the market", *1998 Conference on Computer-Supported Cooperative Work*, Seattle, WA , November 14, 1998, available from http://ccs.mit.edu/klein/ cscw98/paper15

[10] Pei-wu L., Zheng-ding L., "Reduction techniques of workflow verification and its implementation", *Proc. of Int'l Workshop on Grid and Cooperative Computing*, Han Yanbo, Shi Meilin, editors, Beijing, China, 2002, pp703-710

[11] Reichert, M., Dadam, P., "Supporting Dynamic Changes of Workflows without Loosing Control", *Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management*,,Vol.10, No.2, 1998, pp 93-129

[12] Sadiq W., Orlowska M.E., "Analyzing process models using graph reduction techniques", *Information Systems*, Vol.25, No.2, 2000.4, pp117-134

[13] ter Hofstede, A.H.M., Orlowska, M.E., Rajapakse, J. , "Verification problems in conceptual workflow specifications", *Data & Knowledge Engineering*, Vol.24, No.3, 1998, pp.239-256

[14] Towards Adaptive Workflow Systems (CSCW-98 Workshop), 1998 Conference on Computer-Supported Cooperative Work, Seattle, WA , November 14, 1998, available from http://alek.xspaces.org/2005/05/02/cscw98-adaptive-wf-workshops

[15] van der Aalst, W.M.P., "Formalization and verification of event-driven process chains", *Information and Software Technology*, Vol.41, No.10, 1999, pp.639-650

[16] van der Aalst, W.M.P., "Verification of workflow nets", P. Az´ema and G. Balbo, Editors, *Application and Theory of Petri Nets 1997, Proceedings*, Toulouse, France, June 1997, Springer, Berlin, Germany, *Lecture Notes in Computer Science*, Vol.1248, 1997, pp 407–426

[17] Verbeek, H.M.W. , Basten, T. , van der Aalst, W.M.P., "Diagnosing workflow processes using Woflan". Eindhoven University of Technology, Eindhoven, Tech. Rep WP 48, 2000

[18] Weske, M., "Formal foundation and conceptual Design of dynamic adaptations in a workflow management system", *Proceedings of the Thirty Fourth Annual Hawaii International Conference on System Science (HICSS-34)*, R. Sprague, editor, IEEE Computer Society Press, Los Alamitos, California, 2001

[19] Workflow Management Coalition, "The workflow reference model [WfMC1003] WFMC TC00-1003", available from http://www.wfmc.org

Jian Cao

Dr. Jian Cao got his Ph.D degree from Nanjing University of Science & Technology in 2000. He is an associate professor of the department of computer science& engineering, Shanghai Jiaotong University. His main research topics include service computing, cooperative information system and software engineering. He has published more then fifty papers.

Haiyan Zhao

Dr. Haiyan Zhao got his Ph.D degree from Nanjing University of Science & Technology in 2001. She works now in School of Computer Engineering, University of Shanghai for Science and Technology (USST). Before joining in USST, she worked as a research engineer in Shanghai Telecommunication Technology Research Institute..

Jie Wang

Dr. Jie Wang got his Ph.D degree from Stanford University and he is also a consulting faculty of Stanford University now. Dr. Jie Wang conducts research in engineering and environmental informatics. He has also worked for a number of companies and government agencies including Collation, Inc., EDS, Loudcloud, Inc, Stanford University ITSS, Instantis, Inc., First Union Bank, Department of Energy, and NOAA.

Shensheng Zhang

Prof. Shensheng Zhang got his Ph.D degree from Stanford University in 1988. He is a professor of the department of Computer Science& Technology, Shanghai Jiaotong University. His main research topics include distributed computing, agent, virtual reality and software engineering. He is the director of Computer Integration Technology Lab of Shanghai Jiaotong University

Minglu Li

Prof. Minglu Li got his Ph.D degree from Shanghai Jiaotong University in 1996. He is a professor of the department of Computer Science& Technology, Shanghai Jiaotong University. His main research topics include grid computing, image processing, and e-commerce. He is the director of IBM-SJTU Grid Research Center of Shanghai Jiaotong University.