A NEW APPROACH FOR WORKFLOW PROCESS DELTA ANALYSIS BASED ON SYN-NET

Xingqi Huang, Wen Zhao and Shikun Zhang
Peking University, China

Keywords: Delta analysis, workflow, process mining, Syn-net.

Abstract: Many of today's information systems are driven by explicit process models. Creating a workflow design is a complicated process and typically there are discrepancies between the actual workflow processes and the processes as perceived by the management. Delta analysis aims at improving this by comparing process models obtained by process mining from event logs and predefined ones, to measure business alignment of real behaviour of an information system with the expected behaviour. Syn-net is a new workflow model based on Petri-net, with the conceptual foundation synchronizer and suggesting a three-layer perspective of workflow process. In this paper, we propose a new delta analysis approach based on the reduction rules of Syn-net, to examine the discrepancies between the discovered processes and the predefined ones.

1 INTRODUCTION

Workflow is the computerized facilitation or automation of a process, in whole or part where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal (Hollingsworth, 1995).

In a workflow life circle, it consists of four phases: workflow design, workflow configuration, workflow enactment and workflow diagnosis (Aalst et al, 2003). In the traditional approach, well-defined business processes should be designed before enactment is possible and redesigned whenever changes take place. Creating a workflow design is a complicated time-consuming process and typically there are discrepancies between the actual workflow processes and the processes as perceived by the management. In addition, nowadays workflow technology is moving into the direction of more operational flexibility to deal with workflow evolution and workflow exception handling. As a result, participants can deviate from the pre-specified workflow design. Therefore, deviations are natural and the current trend to develop systems allowing for more flexibility accounts for this need.

Clearly one wants to monitor these deviations. For example, a deviation may become common practice rather than being a rare exception. In such a case, the added value of a workflow system becomes questionable and an adaptation is required.

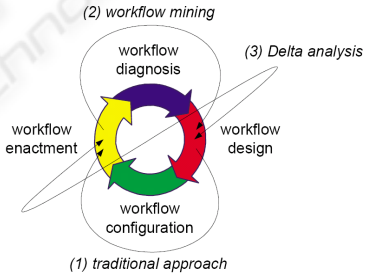

Figure 1: Workflow life circle.

Delta analysis can be applied for analyzing discrepancies between models defined in the design phase and the actual executions registered in the enactment phase. Based on the event log a process model can be derived with process mining technique, reflecting the observed behaviour and therefore providing insight in what actually happened (Aalst et al, 2003; Aalst et al, 2004; Greco et al, 2005). As predefined model specifies how people and organizations are assumed to work, while the discovered one reflects how people and procedure really work, discrepancies between both can be used to improve the process. Also, the current trend in workflow management is to develop systems that allow for even more flexibility. This suggests that it is more interesting to compare predefined process models with “discovered” models.

Huang X., Zhao W. and Zhang S. (2007).
A NEW APPROACH FOR WORKFLOW PROCESS DELTA ANALYSIS BASED ON SYN-NET.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 480-486
DOI: 10.5220/0002370304800486

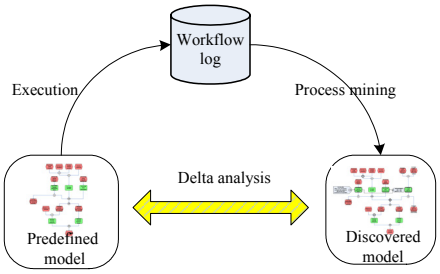

Figure 2: Delta analysis.

For large processes it may be difficult to compare the predefined models and the discovered ones. There are many ways to highlight differences between two models in a graphical fashion. However, most of such approaches will consider simple "node mapping techniques" rather than compare differences in behaviour, i.e., the focus is on syntactical differences rather than semantic differences (Aalst et al, 2005).

In this paper, we propose a folding approach to highlight the discrepancies between the discovered models and the predefine ones, using reduction rules in Syn-net (Yuan, 2005).

The remainder of this paper is organized as follows. Section 2 introduces the Syn-net, which is a synchronization based workflow process model. In section 3, we introduce process mining to discover a process model in Syn-net from event logs, and then present the reduction rules of Syn-net, as well as the folding approach applying the reduction rules. In section 4, we conclude the paper and list the future work.

2 THE SYN-NET

Firstly, let we have a brief introduction to the Syn-net. Here we just explain some most important points of the model. The details of Syn-net are explained rationally in the work of Yuan (2005). Readers can refer to it for more information.

Syn-net is a new workflow model based on Petri-net, with the conceptual foundation synchronizer and suggesting a three-layer perspective of workflow process. In Syn-net, a workflow is defined as the formal description of a business process, including workflow logic that describes dependences between tasks and workflow semantics added on the logic to describe obvious contents, as well as workflow management to check finished tasks according to the workflow logic and start the next task or, to select and start the next task according to the workflow semantics.

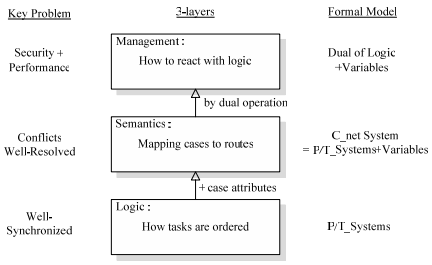

Figure 3: Three layer workflow model.

Workflow logic plays a decisive role in workflow. In general, workflow logic specifies how tasks of a business process are ordered and disordered, i.e. how they are synchronized. The order is derived from the causal dependences among tasks and from organizational regulations. Besides, it covers all possible routes for all possible cases allowed by the business in question, i.e. it is case irrelevant. So it is not concerned with case attributes needed to make decisions on selective routings. Such attributes will be introduced into the logic to form workflow semantics. Furthermore, a task can be executed at most once for each run of the logic. So, iterative routing should not appear as a logic feature. The passing of control from task to task is to be done automatically by workflow engine since control passing involves resource assignment to tasks. The duty of a task is confined to the business itself, not including business management.

The order relation among tasks is defined as follows:

Definition 1 Order Relations among Tasks

All tasks in *TASK* are ordered by the nature of the related business process. So we have a relation $<$, $< \subseteq TASK \times TASK$.

And we have a sub-relation $<\cdot$ of $<$: $<\cdot \subseteq <$ and $(t,t') \in <. \Rightarrow \forall t'' \in TASK : \neg(t < t'' \wedge t'' < t')$.

$<\cdot$ is the next relation among tasks. For $(T_1, T_2) \in <\cdot$ we say that T_2 is immediately before T_2, or T_2 is immediately after T_1. We will define workflow logic by specifying how T_1 and T_2, $T_1 <\cdot T_2$, are synchronized.

The synchronizer is the central concern of workflow logic. A place p with a local structure like in figure 4 is called a synchronizer of pattern (a_1, a_2) between T_1 and T_2 or simply a synchronizer. We write $p = (T_1, T_2, (a_1, a2))$. All places except start and end places are synchronizers in Syn-net.

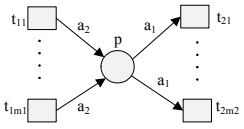

Figure 4: Synchronizer.

In workflow logic, each transition can occur at most once, and a synchronizer p= (T_1, T_2, (a_1, $a2$)) can authorize the post transitions to occur only if M(p)=$a_1 \times a_2$.

Definition 2 Workflow logic

A P/T system $\Sigma = (P,T;F,K,W,M_0)$ is called the workflow logic of (*TASK*, <), or WF_logic for short, if $<\cdot = \{(t,t') \mid t,t' \in T \wedge \exists p \in P : t \in {}^\bullet p \wedge t' \in p^\bullet\}$, $\forall p \in P$: $({}^\bullet p = \phi \Rightarrow M_0(p) = 1) \wedge ({}^\bullet p \neq \phi \Rightarrow M_0(p) = 0)$, and $(T,<\cdot) = (\text{TASK},<\cdot')$.

The workflow semantics on workflow logic denotes how to select a route for a given case based on its attributes. Different from workflow logic, workflow semantics is case-relevant, and it defines a unique route for each case based on case attributes. It specifies how conflicts in workflow logic are resolved with case attributes, involving only those attributes that are needed in making decision on selective routing, while not concerned with actual contents of tasks beyond decision-making attributes. Returns and skips are no need to be introduced into workflow semantics model, for the duty of workflow semantics is only to solve the conflict in the workflow logic and not concerned with the quality of the task that has been done. Whether to redo or not is the concern of workflow management, and the actual return for redoing some tasks is judged by the workflow engine according to the decisions of certain participants or management rules. Such postponed skip and returns are called implicit jumps: jump forward and jump backward, and are dealt with by workflow engine at runtime.

Definition 3 Workflow semantics

A C_net system $\Sigma = (P,V,T;F,K,W,R, W\text{r}, M_\text{T}, M_0)$ is called a workflow semantics, if $(P,T;F,K,W,M_{op})$ is a WF_logic, where $M_{op}=M_0|p$, and $\forall v \in V$: $|w(v)| \leq 1 \wedge |r(v)|>1 \wedge M_0(v)=0$, $\forall t \in T : M_\text{T}=guard\ (t) + body\ (t)$.

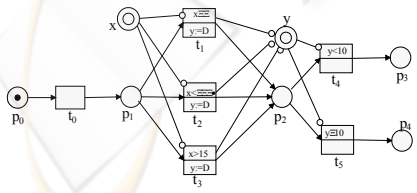

Figure 5: A process model in Syn-net.

Figure 5 is an example of the workflow logic and semantics of a process. Here let we explain the elements in the model. It should be noticed that (P,T;F,K,W) has the same meaning as in P/T systems. And (V, R, Wr, M_T) is introduced from C_net: V is the obvious variables of the *B_form* (Yuan, 2005), and variables in V are used in guard and body of transitions; R is the reading relation between T and V; W_r is the writing relation between T and V; M_T is a marking on transitions, consisting of a guard and a body for $t \in T$; and M_0 is the initial marking of each place in P and variable in V.

The workflow management based on workflow semantic is a formal description of how a workflow engine passes its control from task to task. It resolves conflicts among routes, with given attributes of a specific case, according to the semantics and guards on tasks. In addition, a management system takes care of resource allocation for tasks, time constraint setting and other security matters based on organization-specific database, knowledgebase and rules for management. In Syn-net, the management logic is given in the dual net of workflow logic, for the place in workflow logic is just the management task of workflow engine.

One of the expected properties of a well-designed Syn-net is named throughness.

Definition 4 Throughness

Let $\Sigma = (P,T;F,K,W,M_0)$ be a WF_logic,

1. A marking M reachable from M_0 is a dead marking, if $\forall t \in T : \neg M[t>>$. Let M_D be the set of all dead markings.
2. Let $E = \{p \mid p^\bullet = \varnothing\}$ be the set of all end places of Σ. For each p in E, let M_p be defined as
$$M_p(p') = \begin{cases} 1 & if\ p' = p \\ 0 & if\ p' \neq p \end{cases}$$
For all $p' \in P$, M_p is called the end marking of P.
3. Σ is said to be through if every dead marking is an end marking, i.e. $M_D \subseteq M_E$. We call this property "*Throughness*".

In Syn-net, we have proposed several reduction rules for workflow logic analysis, to prove the throughness of a WF_logic. If a WF_logic can be reduced to a single place by applying the reduction rules, then the WF_logic is dynamically through. For delta analysis, in this paper we will show the rules can also be applied to examine the discrepancies between two Syn-nets.

3 DELTA ANALYSIS

For delta analysis in this paper, the predefined process model must be a Syn-net, as well as the

process model discovered by process mining. So firstly we present a process mining algorithm. The algorithm receives an event log as input and returns a Syn-net as output. It can be seen as an extension of α-algorithm (Aalst et al, 2003; Aalst et al, 2004). And then, we will introduce some reduction rules, and present how to apply these rules to examine the discrepancies between the predefined models and discovered ones.

3.1 Using Process Mining to Obtain a Process Model

Here we present a process mining algorithm to discover a process model in Syn-net from event logs. It can deal with some of the logical problems of α-algorithm such as invisible tasks, short loops, duplicated tasks, and so on. It is assumed that the predefined process model is of throughness. Usually before the workflow process is executed by workflow engine, the model is verified according to some rules. The constraints of Syn-net make it that no such structure as choice and synchronization mixed or synchronization without all its preceding transitions fired.

The logs of workflow engine record events in execution of processes. It is possible to assume that each event refers to an activity, each event refers to a case and it can have a performer, and events can have a timestamp and are totally ordered. Also, the logs contain data elements referring to properties of the case and tasks. Each modification of the data elements is also recorded.

For process mining, we need to discover a Syn-net (P, V, T; F, K, W, R, W_r, M_T, M_0) from the logs. In Syn-net definition, the dependence relation $<\bullet$ is defined as $<\bullet=\{(t, t')|t,t'\in T\wedge \exists p\in P: t\in {}^{\bullet}p\wedge t'\in p^{\bullet}\}$. Here the key is to find the $<\bullet$ relation from logs.

Let T be a set of tasks. Let $\sigma= t_1t_2\ldots t_n\in T^*$ a sequence over T of length n. $\in$, first, and last are defined as follows: 1) $t \in \sigma$ if and only if $t \in \{t_1, t_2, \ldots t_n\}$; 2) if $n \geq 1$, then first(σ) = t_1 and last(σ) = t_n. And next we define order relations between transitions in the log. There are four types of order relations: next, parallel, choice, and only possibility of next. Let L be a log over T, and t_a, $t_b \in T$:

1. $t_a<t_b$: if there is a trace $\sigma = t_1t_2t_3\ldots t_n$, $\sigma\in L$, $t_a=t_i$ and $t_b=t_{i+1}$
2. $t_a||t_b$: if and only if $t_a<t_b$ and $t_a>t_b$
3. $t_a\#t_b$: if neither $t_a<t_b$ nor $t_b<t_a$
4. $t_a<\cdot t_b$: if $t_a<t_b$, and not $t_b<t_a$

From workflow engine's log, we discover relation $<\cdot$ by identifying the next relation $<$ in log, but not in both direction. If $t_a<t_b$, but no $t_b<t_a$ exists, it is very likely that there is a causal dependency or organizational regulation between t_a and t_b.

Let L is a workflow log, and σ is a sequence of transitions in L. The formal description of the algorithm is as follows:

Mining _Process (L):

1. $T_L = \{t \in T \mid \exists_{\sigma\in L}\ t\in \sigma\}$,
2. $T_{first} = \{t \in T \mid \exists_{\sigma\in L}\ t = first(\sigma)\}$,
3. $T_{last} = \{t \in T \mid \exists_{\sigma\in L}\ t = last(\sigma)\}$,
4. $X_L = \{(A,B) \mid A\subseteq T_L \wedge B\subseteq T_L \wedge \forall t_a\in A\ \forall t_b\in B, t_a<\cdot t_b \wedge \forall t_{a1}t_{a2}\in A, t_{a1}\#t_{a2}\wedge \forall t_{b1},t_{b2}\in B, t_{b1}\#t_{b2}\}$,
5. $Y_L = \{(A,B)\in X_L \mid \forall (A',B')\in X_L, A\subseteq A'\wedge B\subseteq B' \Rightarrow (A,B)=(A',B')\}$,
6. $P_{L'} = \{p(t_a, t_b) \mid t_a\in A, t_b\in B \wedge (A,B)\subseteq Y_L\} \cup \{p(null, t_i) \mid t_i\in T_{first}\} \cup \{p(t_o, null) \mid t_o\in T_{last}\}$
7. $F' = \{(t,p) \mid t\in T_L, p\in P_L' \wedge t\in p.pre\} \cup \{((p,t) \mid t\in T_L, p\in P_L' \wedge t\in p.post\}$
$W' = \{(f,1) \mid f\in F'\}$ $K'=\{(p,1) \mid p\in P_L'\}$
8. (P_L,W,K)= Reduce_net(P_L',W',K');
$F = \{(t,p) \mid t\in T_L, p\in P_L \wedge t\in p.pre\} \cup \{((p,t) \mid t\in T_L, p\in P_L \wedge t\in p.post\}$
9. $M_T=\{(t, guard, body) \mid t\in T_L\}$
$V=\{v \mid \exists_{(t,guard,body)\in M_T} v\in guard \vee v\in body\}$
$Wr = \{(t,v) \mid t\in T_L, v\in V \wedge \exists_{mt\in M_T} v\in m_t.body.l\}$
$R = \{(t,v) \mid t\in T_L, v\in V \wedge \exists_{mt\in M_T} v\in m_t.body.r \wedge \exists_{mt\in MT} v\in m_t.guard\}$
$M_0=\{(p,1) \mid p\in \{p(null,t_i) \mid t_i\in T_{first}\} \cup \{(p,0) \mid p\notin \{p(null,t_i) \mid t_i\in T_{first}\}$
10. Syn-net (L) = (P_L, V, T_L, F, K, W, R, W_r, M_T, M_0).

Reduce_net (P_L', W', K'):

Flag=1
While Flag!=0
For each $t_i\in T$, $i=1,2,..,n$
$P_{tempi}=\{p \mid t_i\{p \mid t_i\in p.pre \wedge p\in P_L'\}$
$P^*_i = \{p \mid p.pre = \cup p'.pre \wedge p.post = \cup p'.post \wedge p'\in P_{tempi}\}$
$P_{L1}'=P_L'$; $P_{Li}'=(P_{Li-1}'-P_{tempi}) \cup P^*_i$
$W_{tempi}=\{(f,w) \mid f=(t,p') \wedge t\in \cup p'.pre \wedge p'\in P_{tempi}\} \cup \{(f,w) \mid f=(p',t) \wedge t\in \cup p'.post \wedge p'\in P_{tempi}\}$
$W^*_i=\{(f,w) \mid f=(t,p) \wedge p\in P^*_i \wedge w=\sum w' \wedge (f'',w')\in W_{tempi} \wedge f''\in (t,p') \wedge t\in \cup p'.pre \wedge p'\in P_{tempi}\}$
$W_I'=W'$; $W_i'=(W_{i-1}'-W_{tempi}) \cup W^*_i$
$K_1'=K'$; $K_i'=(K_{i-1}'-\{(p,k) \mid p\in P_{tempi}\}) \cup \{(p,k) \mid p\in P^*i' \wedge k=\sum k,(p,k)\in K_{i-1}', p\in P_{tempi}\}$
$P_L''=P_{Ln}'$; $W''=W_n'$; $.K''=K_n'$
For each $t_i\in T$, $i=n,n-1,\ldots,1$
$P_{tempi}'=\{p \mid t_i\in p.post \wedge p\in P_L''\}$
$P^*_i' =\{p \mid p.pre = \cup p'.pre \wedge p.post = \cup p'.post \wedge p'\in P_{tempi}'\}$
$P_{Ln}''=P_L''$; $P_{Li-1}''=(PLi''-P_{tempi-1}') \cup P^*_{i-1}'$
$W_{tempi}'=\{(f,w) \mid f=(t,p') \wedge t\in \cup p'.pre \wedge p'\in P_{tempi}'\} \cup \{(f,w) \mid f=(p',t) \wedge t\in \cup p'.post \wedge p'\in P_{tempi}'\}$
$W^*_i'=\{(f,w) \mid f=(t,p^*i) \wedge w=\sum w' \wedge (f'',w')\in W_{tempi}' \wedge f''\in (t,p') \wedge t\in \cup p'.pre \wedge p'\in P_{tempi}'\}$
$W_n''=W''$; $W_{i-1}''=(W_i'-W_{tempi-1}') \cup W^*_{i-1}'$
$K_n''=K''$; $K_{i-1}''=K_i''-\{(p,k) \mid p\in P_{tempi-1}'\} \cup \{(p,k) \mid p\in P^*_{i-1}'' \wedge k=\sum k,(p,k)\in K_i'', p\in P_{tempi-1}'\}$

$P_L = P_{L1}''$; $P_L' = P_L$; $W = W_1''$; $W' = W$; $K = K_1''$; $K' = K$
If $P_{tempi} = \emptyset \wedge P_{tempi}' = \emptyset$ then $Flag = 0$;
Return (P_L, W, K);

The Main idea of our algorithm is as follows. Firstly, we define the first and last transitions. Secondly, we construct X_L which is the set of pairs of transitions that have the $<\cdot$ relation, and later refine X_L to Y_L by taking only the largest elements with respect to set inclusion. It assures that the transitions that have choice relation share the common place. Between every pair of transitions, we build a place and arcs connecting the place and transitions. Then the net needs to be reduced, for places in Syn-net need to be synchronizers and the model must meet constrains of Syn-net. It assures that the discovered net has exact amount of places. K and W can be computed with the information recorded in the fourth step and rule of reduction. Later V, W, Wr, R, M_T and M_0 are added onto the logic layer. The process model is discovered in (P, V, T; F, K, W, R, W_r, M_T, M_0).

Table 1: An example of event log.

PIProcessID	TaskID		PIProcessID	TaskID
1001	T1		1004	T3
1002	T1		1002	T8
1003	T1		1003	T4
1001	T2		1003	T5
1002	T3		1004	T4
1002	T2		1001	T6
1001	T3		1002	T9
1003	T3		1004	T5
1001	T4		1003	T7
1004	T1		1003	T9
1003	T2		1001	T9
1002	T4		1004	T7
1001	T5		1004	T9
1004	T2			

With a log in table1, following the mining algorithm we can obtain a process model in Syn-net as in figure 6, which actually describes a business process of Land and Resource Bureau.

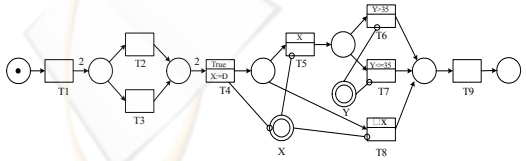

Figure 6: Discovered process model.

It can deal with problems such as invisible tasks and one-length loops. Firstly, because there is no transition special for a routing purpose as in WF-net, there will be no invisible task in our original process model. Therefore, the problem with our process mining work, that invisible task cannot be discovered, may not stand. Secondly, there is no loop or return in the process model presented by Syn-net, for they are the duty of workflow management, not the workflow logic and semantics. However, when the process is at execution time, actual loops may occur, or a task in model may extend to multiple copies to be executed by different participants. Hence the log may contain a sequent of transitions like $\sigma = t_1 t_{2\ldots} . t_k t_k . .t_n,,$ which could be called one-length loop. Obviously, with our model, the one length loop makes no impact on the ability of discovering. Both the original process model and the mined one contain exactly one copy of t_k which is doubled in execution. Consequently, the drawback of α-algorithm that it cannot be dealt with one-length loop is solved.

3.2 Reduction Rules

Now we introduce some reduction rules in Syn-net for delta analysis. These reduction rules can be classified into two groups: one group with rules 1, 2 and 3 that do not impact transition set T, and the other group with rules 4, 5 and 6 that reduce T.

Reduction Rule 1

For $T_1 = \{t_1, \cdots, t_{a_1}\}$, $T_2 = \{t_1', \ldots, t_{a_2}'\}$, we have

(a) the set of synchronizers $\{(\{t_i\}, T, (1, a)) \mid t_i \in T_1$ is reducible to a single synchronizer $(T_1, T, (a_1, a))$ whose capacity is $a_1 \cdot a$;

(b) the set of synchronizers $\{(T, \{t_j'\}, (a, 1)) \mid t_j' \in T_2\}$ is reducible to a single synchronizer $(T, T_2, (a, a_2))$ whose capacity is $a_2 \cdot a$,

where T is a set of tasks with $T_1 <\cdot T$ in (a) and $T <\cdot T_2$ in (b), $1 \le a \le |T|$.

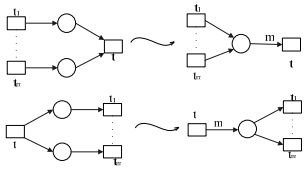

Figure 7: Reduction rule 1.

Reduction Rule 2

Let be $T = \{t_1, \ldots, t_m\}$, $T_i = \{t_{ij} | 1, \ldots, a\}$ for $i = 1, \ldots, m$, and $T_i \cap T_j = \varnothing$ for $i \neq j$, $p_i = (\{t_i\}, T_i, (1, a))$ for all i are AND-split synchronizers, then if T belongs to a single AND synchronizer $(T', T'', (a_1', a_2'))$ with $T \subseteq T''$, $\{p_i | i = 1, \ldots, m\}$ can be reduced to $p = (T, \bigcup T_i, (1, a))$ with $K(p) = m \times a$.

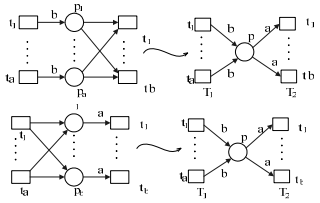

Figure 8: Reduction rule 2.

Reduction Rule 3

Let be $T=\{t_1,\ldots,t_m\}$, $T_i=\{t_{ij}|\ j=1,\ldots,a\}$ for i=1,…,m, and $T_i \cap T_j = \varnothing$ for $i \neq j$, $p_i=(\{t_i\},T_i,(1,a))$ for all i are AND-split synchronizers, then if T belongs to a single AND synchronizer $(T',T'',(a_1',a_2'))$ with $T \subseteq T''$, $\{p_i|i=1,\ldots,m\}$ can be reduced to $p=(T,\cup T_i,(1,a))$ with $K(p)=m \cdot a$.

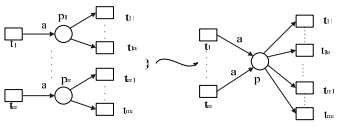

Figure 9: Reduction rule 3.

Reduction Rule 4

Let $p_1=(T_1,T,(a,a_1))$ and $p_2=(T,T_2,(b_1,b))$ be synchronizers and (p_1,T,p_2) is consistent, then p_1 and p_2 can be reduced to synchronizer $p=(T_1,T_2,(a,b))$.

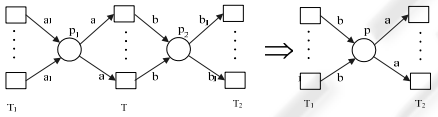

Figure 10: Reduction rule 4.

Reduction Rule 5

If $u^\bullet=\{p\}$ and $^\bullet v=\{p\}$, where $p=(\{u\},\{v\},(1,1))$, then $(\{u\}, p, \{v\})$can be reduced to a single task t with $^\bullet t=^\bullet u$ and $t^\bullet=v^\bullet$.

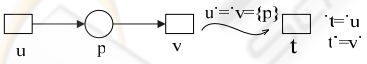

Figure 11: Reduction rules 5.

Reduction Rule 6

If transition t and place p_1, p_2 are satisfied with $t \in p_1^\bullet \wedge \{t\}=^\bullet p_2 \wedge (p_2^\bullet \neq \varnothing \vee p_1^\bullet=\{t\})$, then p_1 and p_2 can be reduced into one place p, $^\bullet p=^\bullet p_1, p^\bullet=p_1^\bullet \cup p_2^\bullet-\{t\}$. If p_1 contains tokens, p also contains tokens.

Figure 12: Reduction rule 6.

3.3 Applying the Rules for Delta Analysis

For large processes it may be difficult to compare the predefined models and the discovered ones. In this paper, we present a folding method for delta analysis using reduction rules in Syn-net.

The central idea of delta analysis in this paper is to fold the identical parts of the two models using reduction rules, so as to highlight the differences between them.

The input of the folding is the predefined process model and discovered one, both of which are in Syn-net. Firstly, with a dynamically through process model A and its execution log, we apply process mining to discover a process model named B in Syn-net. Then, the folding works as follows:

```
Folding:
 while(!stable1 || !stable2)
   while(! stable1)
     Apply rule 1- 3 to A, with transition set N;
     With N of B, if∀t∈N, ∀t'∈N, t in A, t' in B
       corresponding to t in A, has•(t)=r(t') and w(t)=w(t')
     Apply the same rule on B with N;
     if(succ)
           commit reduction to A & B;
           stable1count = 0; stable2 = false;
     else
           undo reduction to A; stable1count++;
     if(stable1count==3) stable1 = true;
   while(!stable2)
     Apply rule 4 - 6 to A, with transition set N;
     With N of B, if∀t∈N, ∀t'∈N, t in A, t' in B
       corresponding to t in A, has•(t)=r(t') and w(t)=w(t')
     Apply the same rule on B with N;
     if(succ)
           commit reduction to A & B;
           stable2count = 0; stable1 = false;
     else
           undo reduce to A; stable2count++;
     if(stable2count==3) stable2 = true;
```

Figure 13: Folding Algorithm.

In the algorithm above, we alternatively apply rules in group 1 and 2 to A and B, and with sequence order for rules within each group, until no more rules can be applied on both of them. Once a rule can be applied on A, if it can also be applied on B with the same transition set N, and for each corresponding pair of transitions t in A and t' in B, $\tau \in N$, $\tau' \in N$, $r(t)=r(t')$ and $w(t)=w(t')$, then commit the reduction on both A and B, with nodes after reduction replace the original ones, else the impact of reduction on A must be withdrew. Then try other rules in the group and later the other group, until

reach the fixed point, where no more rules can be applied on both the nets. As a result, the identical parts of both the two nets are folded, with discrepant parts of the two nets left. If the two models are entirely the same as each other, both of them will be reduced to a single place, for we assume they are of throughness.

In fact, the reduction is to ignore details unconcerned. The same parts of the two nets are of no importance in redesign of the business process, because the actual behaviour is the same as expected.

For example, figure 14 is the predefined process model *A*, and figure 15 is the discovered one named *B*. There is only one difference between them: in *A* the transition t_8 is or-split with t_5, and t_8 is or-split with t_6 and t_7 in *B*.

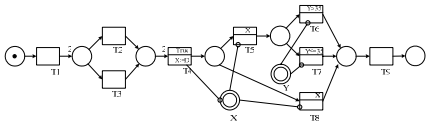

Figure 14: Predefined process model.

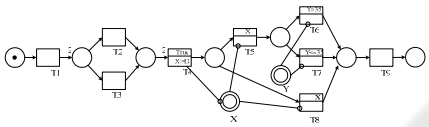

Figure 15: Discovered process model.

Firstly the rule 4 can be applied on A, and the rule can also be applied on B, then reduction of both nets is committed, transitions t2, t3 and their connected places are replaced by a single place. Later, reduction rules can still be applied on A to reduce it to a single place. However, because of the discrepancy between A and B, no more rules can be applied on both A and B for the same transition set, leaving structures highlighting the discrepancies, as shown in figure 16.

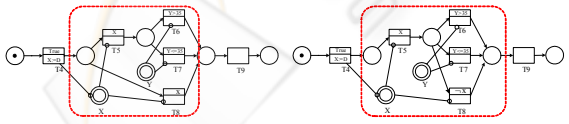

Figure 16: Folding.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we discussed topics of workflow diagnosis phase in workflow life circle.

For delta analysis, firstly a process mining is applied on the event logs to discover a process model. Syn-net is a new workflow model with the conceptual foundation synchronizer and suggesting a three-layer perspective of workflow process. We presented a process mining algorithm extending α-algorithm, with the assumption that the predefined model is in Syn-net and of throughness. Because of the characteristic of Syn-net, following the algorithm some of the drawbacks can be solved.

Since we use a folding approach for delta analysis, we introduced some reduction rules for folding. These rules specify how to find and fold the identical structures of the predefined model and the discovered one. We also presented a folding algorithm to apply these rules, so as to highlight the discrepancies between process models.

However, this work is far from being complete. Since Syn-net is a new workflow model, more researches are needed to refine the theory. For delta analysis, reduction rules may need to be refined and maybe some other rules will be added. Also, the mining and folding algorithms are to be optimized to put them into practice. Our future work will focus on these aspects.

REFERENCES

David Hollingsworth., 1995. The Workflow Reference Model*, Document Number TC00-1003[S]*, Workflow Management Coalition

W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G.Schimm, and A.J.M.M. Weijters, 2003. Workflow Mining: A Survey of Issues and Approaches, *Data & Knowledge Engineering,* Volume 47, Issue 2, 237-267. Elsevier.

W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, 2004. Workflow Mining: Discovering Process Model from Event Logs, *IEEE Transaction on Knowledge and Data Engineering*, 16(9), 1128-1142

Gianluigi Greco, Antonella Guzzo, Giuseppe Manco and Domenico Sacca, 2005. Mining and Reasoning on Workflows. *Knowledge and Data Engineering, IEEE Transactions*, Volume17, Issue 4, 519-534

W.M.P. van der Aalst, 2005. Business alignment: using process mining as a tool for Delta analysis and conformance testing. *Requirement Engineering* 10(3), 198-211

Chongyi Yuan, 2005. *Principals and Application of Petri Nets.* Publishing House of Electronics Industry. 213-258.