Evaluation Method of Change Time for Migrate Dynamic Workflow Changes

Akira Mishima , Shingo Yamaguchi, Minoru Tanaka,
Faculty of Engineering, Yamaguchi University
2-16-1 Tokiwadai, Ube, 755-8611, Japan

Qi-Wei Ge
Faculty of Education, Yamaguchi University
1677-1 Yoshida, Yamaguchi, 753-8513, Japan

Abstract

A workflow is a business process automated by computers. The instance of workflow is called case. Companies need to refine their current workflows in order to adapt them to meet various requirements. The change of a current workflow is called *dynamic change* of the workflow. Ellis et al. have proposed three change types, *Flush*, *Abort*, and *Synthetic Cut-Over (SCO)*, in 1995. As a remarkable change type, Sadiq et al. has proposed *Migrate*. Migrate is a change type that changes the workflow definition immediately, and minimizes the reprocessing of cases. A workflow can be modeled as a Petri net, called workflow net. The workflow net model of Migrate has been proposed. Furthermore the method of transferring cases from an old workflow to a new workflow is also proposed. In this paper, we are to do the performance evaluation on *change time* for dynamic changes of various and complex workflow nets by Migrate. Change time is a measure for evaluating dynamic changes quantitatively. We first propose operations to change workflow nets, called *change-operations*, and a method of generating various and complex workflow nets by using the change-operations. Then we propose a method of computing change time for dynamic changes by Migrate. Applying the computation method, we do the performance evaluation on change time for dynamic changes of 270 generated workflow nets by Migrate. Furthermore we compare Migrate with Ellis et al.'s three change types.

Keywords workflow, dynamic change, Migrate change, change time, Petri nets

1. Introduction

In most of office businesses, information is generally computerized and handled through computer networks in recent years. As a core technology to support the businesses in next generation, *workflow management systems* [1] have been receiving increasing attention. A workflow is a business process automated by computers, and workflow management systems are systems to realize managing and automating the flow of work. The instance of workflow is called case. Introducing the workflow management system has the advantage of reduction of waiting time, prevention of human error and work progress that is easily understood.

In order to adapt to new technology, demand in market, and amended laws, companies need to continually refine their workflows. To change the current workflows with the system running is called *dynamic change* of the workflow. In the current environments changing quickly, it is very important factor to change workflows dynamically. There are cases in change region when a dynamic change of a workflow is required, then it causes a troublesome problem: how do we handle the cases? Dynamic change may cause *dynamic change bugs* [2], which are errors caused by dynamic change which included some tasks being skipped. It is necessary to use the dynamic change type that does not cause dynamic change bug.

Some studies have been addressed on the dynamic change of workflows [3,4,5,6]. Ellis et al. [3] proposed three types of dynamic changes,

Flush, *Abort*, and *Synthetic Cut-Over (SCO)* that are modeled by Petri nets and keep consistency of workflows in 1995. Performance evaluations on change time [7] and transient time [8], which are measures of evaluating dynamic changes quantitatively, have been done to these three dynamic change types, and the following results are obtained.

- SCO is the best of them all.
- Abort is better than Flush unless the arrival interval of input cases is short.

As a remarkable change type, Sadiq et al. [4] proposed *Migrate* dynamic change. Migrate is a change type that changes the workflow definitions immediately, and minimizes redoing works for running cases. Sadiq et al. gave the concept of Migrate, but did not give a formal model of Migrate. Then, in [5], we proposed the formal model of Migrate and how to map cases from the old workflow to the new workflow. Here, it is necessary to do the performance evaluation of Migrate. It is our purpose to do the performance evaluation of Migrate on change time and compare Migrate change type with Ellis et al.'s change types.

To do the performance evaluation, we need complex and various types of examples of dynamic changes. In this paper, we propose a method of generating such workflow nets after change. First we propose basic operations to change workflow nets, called *change-operations*. Then we apply them to change old workflow net to new workflow net. After that we present a method of computing change time of Migrate based on the already proposed Petri net model. Finally we apply these methods to 270 examples generated by using change-operations, and show experimental results. From the experimental results, we evaluate Migrate dynamic change by comparing with Ellis et al.'s three change types.

2. WF-Nets and Dynamic Changes

A workflow is a business process automated by computers. Plural people are engaged in the work, and flow of work is a series, parallel, or mixed structure of them. A unit of work within workflow is called an *activity*, and an instance running in accordance with the definition of workflow is called a *case*. In general, there are many cases which are handled according to the same workflow definition. These cases are handled in the order of *First-In First-Out* (FIFO).

2.1 Workflow Net

A workflow modeled by Petri net is called *workflow net* (WF-net) [9]. In this paper, we use extended WF-net by introducing time in order to do time-dependent analysis. The extended WF-net is defined as follows:

Definition 1: A timed WF-net is a timed Petri net $TPN = (P, T, A, D)$ which satisfies the following:

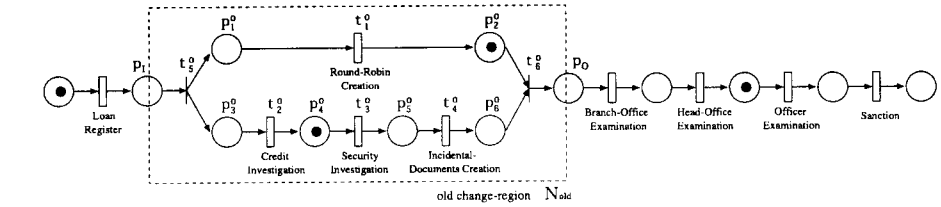
(i) P is a set of places, T is a set of transitions, A is a set of arcs, and D is a set of the delay-time of transitions. Furthermore, δ is a mapping from each transition t_i to delay-time d_i.

(ii) TPN has one input (source) place p_I and one output (sink) place p_O.

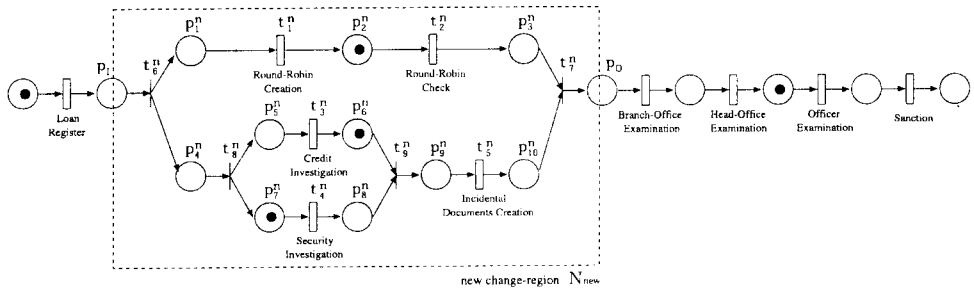
(iii) Every place and transition is located on a path from p_I to p_O.

In WF-nets, a place, a transition, an arc, and a delay-time of transition represent state of process, an activity, a flow relation, and the processing time of the activity, respectively. Further, a token represents a case. To simplify our performance evaluation, this paper assumes that WF-nets are acyclic marked graphs, and the arrival interval d^* of input cases is constant and not less than $\max_{d_i \in D}\{d_i\}$. To express branch and merging of process, in this paper, we introduce *split transition* and *join transition* which have no delay-time, respectively. Split and join transitions are denoted by bars "|" in figures.

As an example, we show the WF-net model of the workflow for the bank loan business process, shown in [10], in Fig.1(a). When a customer applies for a loan, a new case is generated. After that, a person in charge checks the contents, and registers the case. Then the round robin for the case is created, and the incidental documents are created based on credit investigation and security investigation of an applicant in parallel. Then a branch office, a head office, and officer



(a) old WF-net



(b) new WF-net

Figure 1. Dynamic change of the workflow for the bank loan business process

examination perform based on the round robin and the incidental documents, and the case is approved.

2.2 Dynamic Change of Workflows

In terms of WF-nets, a dynamic change is to replace a subnet $N_{old} = (P^o, T^o, A^o, D^o)$ in the original net with a new subnet $N_{new} = (P^n, T^n, A^n, D^n)$, which results in a new net [7]. N_{old} is called *old change-region*, N_{new} is called *new change-region*. In N_{new} and N_{old}, places, transitions, arcs, and delay-times are generally different, but p_I and p_O are common. The change types treated in this paper are as follows:

(1) Flush

N_{old} is replaced by N_{new} later after all tokens in N_{old} are outputted to p_O. Tokens newly arriving at p_I are kept waiting until the replacement of N_{old} by N_{new} finishes.

(2) Abort

N_{old} is immediately replaced by N_{new}, and all tokens in N_{old} are put back to p_I. Then all tokens including newly arriving ones are handled in N_{new}.

(3) SCO (Synthetic Cut-Over)

N_{new} is immediately added to the original net without removing N_{old}. Tokens newly arriving at p_I can only go through N_{new}. N_{old} is removed after all tokens on N_{old} are outputted to p_O. Therefore, after the dynamic change is required, both N_{old} and N_{new} exist for some time. Further, the tokens handled according to N_{new} cannot be outputted to p_O until all tokens in N_{old} are outputted to p_O, because all tokens are handled in the order of FIFO.

(4) Migrate

N_{old} is immediately replaced by N_{new}. The transitions which fired in N_{old} are treated as fired in N_{new} and cases are transferred to the suitable places of N_{new}, which make redoing the activities minimum. Then all tokens including newly arriving ones are handled in N_{new} [5].

As an example, we consider a dynamic change of the workflow for the bank loan business process mentioned above. Here we suppose that the bank decides to add an activity of checking round robin after creating and to perform credit investigation and security investigation in parallel. Figure 1(b) shows the WF-net model after the dynamic change of the workflow.

2.3 Change Time

When dynamic change is required, there most probably exists tokens in the change-region N_{old}. These tokens must be handled according to either N_{old} or N_{new}. Because all tokens must be handled in the order of FIFO, the waiting time occurs before the token newly arriving at p_I is outputted to p_O after dynamic change is required. Such a waiting time is called *change time* [7]. In order to simplify our performance evaluation, we assume that time needed for the removal of N_{old} and the addition of N_{new} is 0, because these times are the same for any dynamic changes. The definition of change time is given as follows:

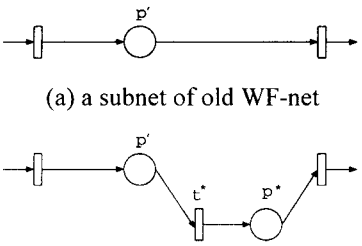
Definition 2: Let τ_{p_I}, τ_{p_O}, and $\Delta\tau_{N_{new}}$ be time when the first token is inputted to p_I after a dynamic change is required, time when the token is outputted to p_O, and the minimum period for a token to move from p_I to p_O in N_{new} with no waiting time, respectively. Change time γ is given by $\gamma = (\tau_{p_O} - \tau_{p_I}) - \Delta\tau_{N_{new}}$.

3. Generating Changed WF-nets

In this section, we propose a method of generating examples of dynamic change of WF-nets in order to do the evaluation experiment. In our experiments, we should have more complex and various changes for an example of dynamic change of workflow. Changing WF-net, in general, we apply some operations to old WF-net, and obtain new WF-net. Here, we call a set of basic operations added to old WF-net *change-operations*, and propose change-operations by which most changes of workflows can be expressed in combinations repeatedly.

The primitive operations applied to workflow nets are only four: addition of node (place or transition), deletion of node, addition of arc, and deletion of arc. Any change can be realized by repeatedly combining the primitive operations. As the operations that may be frequently applied to an old workflow, we propose change- operations that are taken in the view of series and parallel and are combined with these primitive operations. The change-operations proposed in this paper are as follows:

Op1: Sequential insertion of activity.
Op2: Parallel insertion of activity.



(a) a subnet of old WF-net

(b) the new subnet

Figure 2. Sequential insertion of activity.

Op3: Deletion of activity.
Op4: Replacement of activity.
Op5: Sequential exchange of activities.
Op6: Parallel exchange of activities.
Op7: Parallelization of activities.
Op8: Serialization of activities.
Op9: Addition of order-relation between activities.
Op10:Deletion of order-relation between activities.

Change of most workflows can be realized by combinatively applying the above change-operations repeatedly. Of course, after applying change-operations, the changed nets must keep the consistency of WF-nets. Because WF-nets after applying change-operations must be acyclic marked graphs, split transitions and join transitions such as or ($|t^{\bullet}| \geq 2$ or $|{}^{\bullet}t| \geq 2$) must be excepted from target of applying operations.

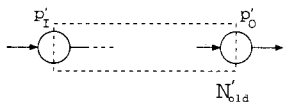
We apply change-operations to an old WF-net $N_{old} = (P^o, T^o, A^o, D^o)$, and obtain a new WF-net $N_{new} = (P^n, T^n, A^n, D^n)$. The formal definitions of each change-operation are given in the following. Here, examples of applying the change-operations are also illustrated.

(1) Sequential insertion of activity

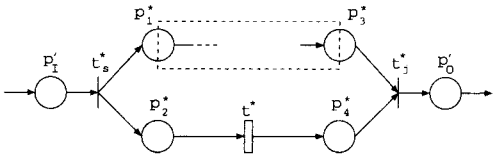
"Sequential insertion of activity" is to choose one place p' in N_{old}, and connect new transition t^* behind p' if $p' \neq p_O$, otherwise connect t^* in front of p'. The formal definition is as follows.

Definition 3: Let p', t^* be any place in N_{old}, transition added newly, respectively. After applying "Sequential insertion of activity", N_{new} is as follows. (See Fig.2)

(i) if $p' \neq p_O$,
$P^n = P^o \cup \{p^*\}$,
$T^n = T^o \cup \{t^*\}$,



(a) a subnet of old WF-net



(b) the new subnet

Figure 3. Parallel insertion of activity.

$$A^n = A^o \cup \{(p',t^*),(t^*,p^*)\} \cup \{(p^*,t) \mid t \in p'^\bullet\} - \{(p',t) \mid t \in p'^\bullet\},$$
$$D^n = D^o \cup \{d^*\}, \quad d^* = \mathcal{D}(t^*).$$

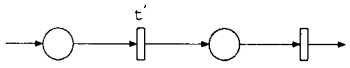
(ii) if $p' = p_O$,
$P^n = P^o \cup \{p^*\}$,
$T^n = T^o \cup \{t^*\}$,
$$A^n = A^o \cup \{(p^*,t^*),(t^*,p_O)\} \cup \{(t,p^*) \mid t \in {}^\bullet p_O\} - \{(t,p_O) \mid t \in {}^\bullet p_O\},$$
$$D^n = D^o \cup \{d^*\}, \quad d^* = \mathcal{D}(t^*).$$

(2) Parallel insertion of activity
"Parallel insertion of activity" is to connect new transition t^* to sub-WF-net N'_{old} in parallel. The formal definition is as follows.

Definition 4: Let N'_{old}, t^* be any subnet in N_{old}, transition added newly, respectively. Further, p'_I, p'_O is the input place, the output place of N'_{old}, respectively. After applying "Parallel insertion of activity", N_{new} is as follows. (See Fig.3)

$$P^n = P^o \cup \{p_1^*, p_2^*, p_3^*, p_4^*\},$$
$$T^n = T^o \cup \{t^*, t_s^*, t_j^*\},$$
$$A^n = A^o \cup \{(p'_I,t_s^*),(t_s^*,p_1^*),(t_s^*,p_2^*),(p_2^*,t^*),(t^*,p_4^*),(p_3^*,t_j^*),(p_4^*,t_j^*),(t_j^*,p'_O)\} \cup \{(p_1^*,t) \mid t \in p_I'^\bullet\} \cup \{(t,p_3^*) \mid t \in {}^\bullet p'_O\} - \{(p'_I,t) \mid t \in p_I'^\bullet\} - \{(t,p'_O) \mid t \in {}^\bullet p'_O\},$$
$$D^n = D^o \cup \{d^*, d_s^*, d_j^*\}, \quad d^* = \mathcal{D}(t^*),$$
$$d_s^* = \mathcal{D}(t_s^*) = 0, d_j^* = \mathcal{D}(t_j^*) = 0.$$

(3) Deletion of activity
"Deletion of activity" is to delete any transition t' in N_{old}. The formal definition is as follows.



(a) a subnet of old WF-net

(b) the new subnet

Figure 4. Deletion of activity.

Definition 5: Let t' be any transition in N_{old}. After applying "Deletion of activity", N_{new} is as follows. (See Fig.4)

(i) if $p_O \notin t'^\bullet$,
$P^n = P^o - t'^\bullet$,
$T^n = T^o - \{t'\}$,
$$A^n = A^o \cup \{(p,t) \mid p \in {}^\bullet t', t \in (t'^\bullet)^\bullet\} - \{(p,t') \mid p \in {}^\bullet t'\} - \{(t',p) \mid p \in t'^\bullet\} - \{(p,t) \mid p \in t'^\bullet, t \in (t'^\bullet)^\bullet\},$$
$$D^n = D^o - \{d'\}, \quad d' = \mathcal{D}(t').$$

(ii) if $p_O \in t'^\bullet$,
$P^n = P^o - {}^\bullet t'$,
$T^n = T^o - \{t'\}$,
$$A^n = A^o \cup \{(t,p_O) \mid t \in {}^\bullet({}^\bullet t')\} - \{(t,p) \mid t \in {}^\bullet({}^\bullet t'), p \in {}^\bullet t'\} - \{(p,t') \mid p \in {}^\bullet t'\} - \{(t',p_O)\},$$
$$D^n = D^o - \{d'\}, \quad d' = \mathcal{D}(t').$$

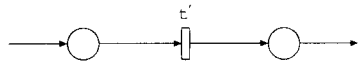
(4) Replacement of activity
"Replacement of activity" is to replace any transition t' in N_{old} with new transition t^*. The formal definition is as follows.

Definition 6: Let t', t^* be any transition in N_{old}, new transition added instead of t', respectively. After applying "Replacement of activity", N_{new} is as follows. (See Fig.5)

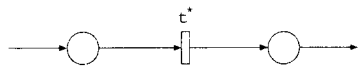
$$P^n = P^o,$$
$$T^n = T^o \cup \{t^*\} - \{t'\},$$
$$A^n = A^o \cup \{(p,t^*) \mid p \in {}^\bullet t'\} \cup \{(t^*,p) \mid p \in t'^\bullet\} - \{(p,t') \mid p \in {}^\bullet t'\} - \{(t',p) \mid p \in t'^\bullet\},$$
$$D^n = D^o \cup \{d^*\} - \{d'\}, d^* = \mathcal{D}(t^*), d' = \mathcal{D}(t').$$

(5) Sequential exchange of activities

"Sequential exchange of activities" is to replace the positions of two transitions t_1' and t_2' in N_{old} which have order relation. The formal definition is as follows.

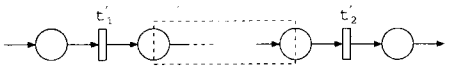


(a) a subnet of old WF-net

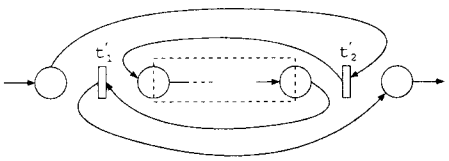


(b) the new subnet

Figure 5. Replacement of activity.



(a) a subnet of old WF-net



(b) the new subnet

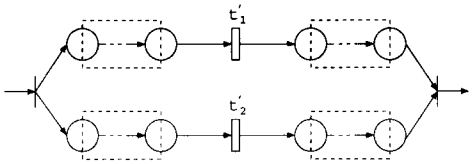
Figure 6. Sequential exchange of activities.

Definition 7: Let t_1' and t_2' be any transitions in N_{old}. Here, t_1' and t_2' are different transitions and there exists at least one path from t_1' to t_2' or from t_2' to t_1'. After applying "Sequential exchange of activities", N_{new} is as follows. (See Fig.6)

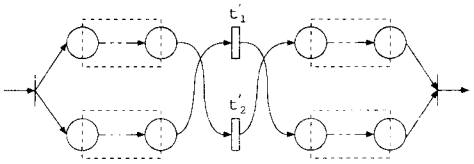
$$
\begin{aligned}
&P^n = P^o,\\
&T^n = T^o,\\
&A^n = A^o \cup \{(p,t_2') \mid p \in {}^\bullet t_1'\} \cup \{(t_2',p) \mid p \in t_1'^\bullet\}\\
&\quad \cup \{(p,t_1') \mid p \in {}^\bullet t_2'\} \cup \{(t_1',p) \mid p \in t_2'^\bullet\}\\
&\quad - \{(p,t_1') \mid p \in {}^\bullet t_1'\} - \{(t_1',p) \mid p \in t_1'^\bullet\}\\
&\quad - \{(p,t_2') \mid p \in {}^\bullet t_2'\} - \{(t_2',p) \mid p \in t_2'^\bullet\},\\
&D^n = D^o.
\end{aligned}
$$

(6) Parallel exchange of activities

"Parallel exchange of activities" is to replace the positions of two transitions t_1' and t_2' in N_{old} which have no order relation. The formal definition is as follows.



(a) a subnet of old WF-net



(b) the new subnet

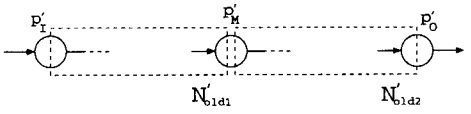
Figure 7. Parallel exchange of activities.

Definition 8: Let t_1' and t_2' be any transitions in N_{old}. Here, t_1' and t_2' are different transitions and there exists no path from t_1' to t_2' and from t_2' to t_1'. After applying "Parallel exchange of activities", N_{new} is as follows. (See Fig.7)

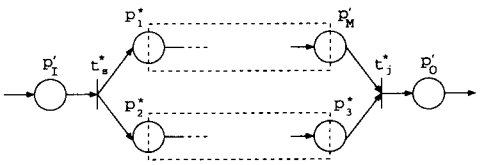
$$
\begin{aligned}
&P^n = P^o,\\
&T^n = T^o,\\
&A^n = A^o \cup \{(p,t_2') \mid p \in {}^\bullet t_1'\} \cup \{(t_2',p) \mid p \in t_1'^\bullet\}\\
&\quad \cup \{(p,t_1') \mid p \in {}^\bullet t_2'\} \cup \{(t_1',p) \mid p \in t_2'^\bullet\}\\
&\quad - \{(p,t_1') \mid p \in {}^\bullet t_1'\} - \{(t_1',p) \mid p \in t_1'^\bullet\}\\
&\quad - \{(p,t_2') \mid p \in {}^\bullet t_2'\} - \{(t_2',p) \mid p \in t_2'^\bullet\},\\
&D^n = D^o.
\end{aligned}
$$

(7) Parallelization of activities

"Parallelization of activities" is to connect in parallel two sub-WF-nets connected in series N'_{old1} and N'_{old2} in N_{old}. The formal definition is as follows.



(a) a subnet of old WF-net

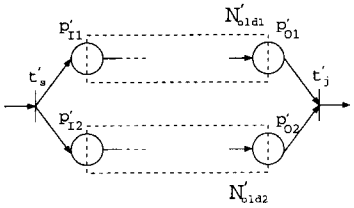


(b) the new subnet

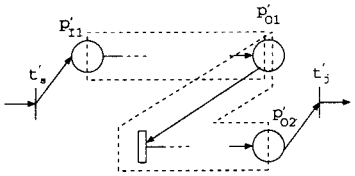
Figure 8. Parallelization of activities.

Definition 9: Let N'_{old1} and N'_{old2} be any sub-WF-net in N_{old} so that output place of N'_{old1} and input place of N'_{old2} are the same place, denoted as p'_M. Furthermore, let p'_I and p'_O be input place of N'_{old1} and output place of N'_{old2}, respectively. After applying "Parallelization of activities", N_{new} is as follows. (See Fig.8)

$$P^n = P^o \cup \{p_1^*, p_2^*, p_3^*\},$$
$$T^n = T^o \cup \{t_s^*, t_j^*\},$$
$$A^n = A^o \cup \{(p'_I, t_s^*), (t_s^*, p_1^*), (t_s^*, p_2^*), (p'_M, t_j^*),$$
$$(p_3^*, t_j^*), (t_j^*, p'_O)\} \cup \{(p_1^*, t) \mid t \in p'^{\bullet}_I\}$$
$$\cup \{(p_2, t) \mid t \in p'^{\bullet}_M\} \cup \{(t, p_3^*) \mid t \in {}^{\bullet}p'_O\}$$
$$-\{(p'_I, t) \mid t \in p'^{\bullet}_I\} - \{(p'_M, t) \mid t \in p'^{\bullet}_M\}$$
$$-\{(t, p'_O) \mid t \in {}^{\bullet}p'_O\},$$
$$D^n = D^o \cup \{d_s^*, d_j^*\},$$
$$d_s^* = \mathcal{D}(t_s^*) = 0, \quad d_j^* = \mathcal{D}(t_j^*) = 0.$$



(a) a subnet of old WF-net



(b) the new subnet

Figure 9. Serialization of activities.

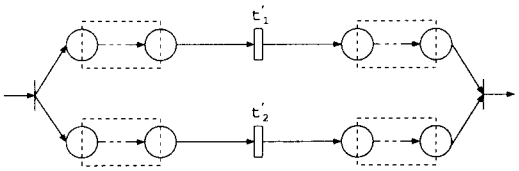
(8) Serialization of activities

"Serialization of activities" is to connect in series two sub-WF-nets connected in parallel N'_{old1} and N'_{old2} in N_{old}. The formal definition is as follows.

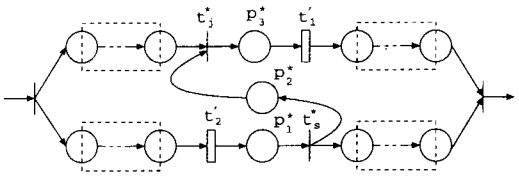
Definition 10: Let N'_{old1} and N'_{old2} be any sub-WF-nets in N_{old} connected in parallel. Here, let p'_{I1}, p'_{I2} be input place of N'_{old1}, N'_{old2}, and p'_{O1}, p'_{O2} be output place of N'_{old1}, N'_{old2}, respectively. Furthermore, let t'_s, t'_j be split transition, join transition which connect N'_{old1} and N'_{old2} in parallel, respectively. After applying "Serialization of activities", N_{new} is as follows. (See Fig.9)

$$P^n = P^o - \{p'_{I2}\},$$
$$T^n = T^o,$$
$$A^n = A^o \cup \{(p'_{O1}, t) \mid t \in p'^{\bullet}_{I2}\}$$
$$-\{(t'_s, p'_{I2}), (p'_{O1}, t'_j)\}$$
$$-\{(p'_{I2}, t) \mid t \in p'^{\bullet}_{I2}\},$$
$$D^n = D^o.$$

After applying "Serialization of activities", if $|{}^{\bullet}t'_s| = |t'^{\bullet}_s| = 1$, $|{}^{\bullet}t'_j| = |t'^{\bullet}_j| = 1$, then apply "Deletion of activity" to t'_s, t'_j to reduction, respectively.



(a) a subnet of old WF-net



(b) the new subnet

Figure 10. Addition of order-relation between activities.

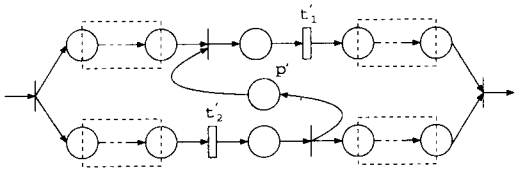
(9) Addition of order-relation between activities

"Addition of order-relation between activities" is to add order-relation to two transitions t'_1, t'_2 in N_{old} that t'_1 cannot fire if t'_2 doesn't fire. The formal definition is as follows.

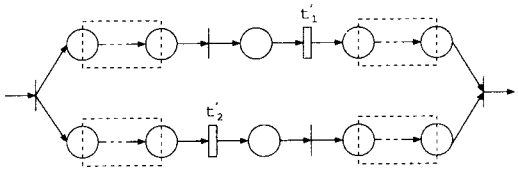
Definition 11: Let t'_1 and t'_2 be any transitions in N_{old}. Here, t'_1 and t'_2 are different transitions and there exists no path from t'_1 to t'_2 and from t'_2 to t'_1. After applying "Addition of order-relation between activities", N_{new} is as follows. (See Fig.10)

$$P^n = P^o \cup \{p_1^*, p_2^*, p_3^*\},$$
$$T^n = T^o \cup \{t_s^*, t_j^*\},$$
$$A^n = A^o \cup \{(t'_2, p_1^*), (p_1^*, t_s^*), (t_s^*, p_2^*), (p_2^*, t_j^*),$$
$$(t_j^*, p_3^*), (p_3^*, t'_1)\} \cup \{(t_s^*, p) \mid p \in t'^{\bullet}_2\}$$

$$\cup\{(p,t_j^*) \mid p \in {}^\bullet t_1'\} - \{(t_2', p) \mid p \in t_2'^\bullet\}$$
$$-\{(p,t_1') \mid p \in {}^\bullet t_1'\},$$
$$D^n = D^o \cup \{d_s^*, d_j^*\},$$
$$d_s^* = \mathcal{D}(t_s^*) = 0, \quad d_j^* = \mathcal{D}(t_j^*) = 0.$$



(a) a subnet of old WF-net



(b) the new subnet

Figure 11. Deletion of order-relation between activities.

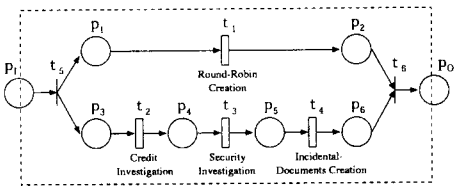
(10) Deletion of order-relation between activities

"Deletion of order-relation between activities" is to delete order-relation to two transitions t_1', t_2' in N_{old} that t_1' cannot fire if t_2' doesn't fire. The formal definition is as follows.

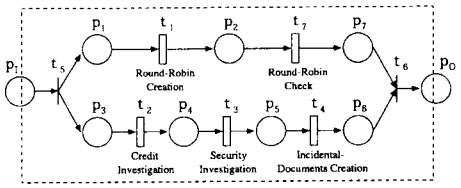
Definition 12: Let t_1' and t_2' be any transitions in N_{old} there exists a place p' whose input transition is a split transition and output transition is a join transition. After applying "Deletion of order-relation between activities", N_{new} is as follows. (See Fig.11)

$$P^n = P^o - \{p'\},$$
$$T^n = T^o,$$
$$A^n = A^o - \{(t,p') \mid t \in {}^\bullet p'\}$$
$$-\{(p',t) \mid t \in p'^\bullet\},$$
$$D^n = D^o.$$

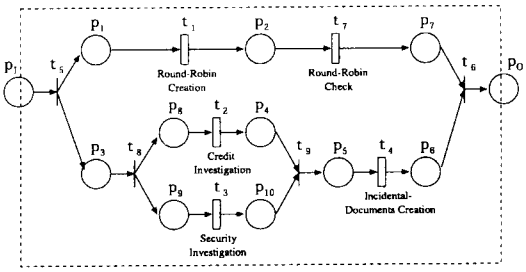
After "Deletion of order-relation between activities", if $|{}^\bullet({}^\bullet p')| = |({}^\bullet p')^\bullet| = 1$, $|{}^\bullet(p'^\bullet)| = |(p'^\bullet)^\bullet| = 1$, then apply "Deletion of activity" to ${}^\bullet p'$, $p'^\bullet$ to reduction, respectively.



(a) old change region.



(b) change region after applying "Sequential insertion of activity".



(c) change region after applying "Paralleization of activities".

Figure 12. An example of applying change-operations.

Here we give an example of applying change-operations to a WF-net. We consider a dynamic change of the workflow for the bank loan business process shown in Fig.1. The old change region is shown in Fig.12(a). First, we apply "Sequential insertion of activity" to the old WF-net. We insert new activity "Round- Robin Check" after "Round-Robin Creation". The WF-net after applying "Sequential insertion of activity" is shown in Fig.12(b). Then we apply "Parallelization of activities" to the changed WF-net. We connect in parallel "Credit Investigation" and "Security Investigation". The WF-net generated by applying "Parallelization of activities" is shown in Fig.12(c). Thus, dynamic

change of the workflows can be realized by combinatively applying the change-operations repeatedly.

4. Computation of Change Time of Migrate

The change time of Flush and Abort can be analytically calculated by a formula [7]. Concerning SCO, the computing method by algorithm has been proposed because it is difficult to compute change time by a formula [7]. Similarly as SCO, it is difficult to compute change time of Migrate by a formula due to the complicated markings generated by Migrate. Therefore, we propose an algorithm to compute change time of Migrate.

In order to compute the change time of Migrate $\gamma_{migrate}$, we use the formula of Definition 2. For τ_{p_O}, we propose the following algorithm.

```
⟨⟨ Computation of τ_{P_O} ⟩⟩
Procedure Compute_τ_{P_O}
/* Input:
  N_new : new change-region;
  M^n :   marking of N_new ;
  T_F :   set of transitions having decided
          to fire;
  R :     array of remain-times until fire;
  τ_cs :  start time of dynamic change;
  d* :    arrival interval of input cases;
  Output: τ_{P_O} */
begin
/* 1° Initialization of τ, τ_{p_I} . */
  τ ← τ_cs ; τ_{p_I} ← ⌈τ_cs / d*⌉ d*;
/* 2° Repeat the followings until the
        first token arrived N_new is
        outputted to p_O . */
  while ((τ_{p_I} > τ) or
          (Σ_{p^n ∈ ρ^n − {p_O}} M^n(p^n) > 0)) do
    begin
    /* 2.1° Fire transitions for
              non-activities. */
      for i ← 1 to |T^n| do
        if ((d_i^n = 0) and
            (∀p ∈ •t_i^n, M^n(p) > 0)) then
          begin
            ∀p ∈ •t_i^n, M^n(p) ← M^n(p) − 1;
            ∀p ∈ t_i^n •, M^n(p) ← M^n(p) + 1;
          end
    /* 2.2° Fire transitions for activities.*/
        for i ← 1 to |T^n| do
          begin
            if ((t_i^n ∉ T_F) and
              (∀p ∈ •t_i^n, M^n(p) > 0)) then
              begin
                R[t_i^n] ← d_i^n ;
                T_F ← T_F ∪ {t_i^n};
              end
            if R[t_i^n] = 0 then
              begin
                ∀p ∈ •t_i^n, M^n(p) ← M^n(p) − 1;
                ∀p ∈ t_i^n •, M^n(p) ← M^n(p) + 1;
                T_F ← T_F − {t_i^n};
              end
          end
    /* 2.3° Input the first token newly
            arriving at p_I after τ_cs . */
      if τ = τ_{p_I} then
          M^n(p_I) ← M^n(p_I) + 1;
    /* 2.4° Update τ . */
      if ((τ_{p_I} > τ) or
          (Σ_{p^n ∈ ρ^n − {p_O}} M^n(p^n) > 0)) then
        begin
          τ ← τ + 1;
          ∀t^n ∈ T_F, R[t^n] ← R[t^n] − 1;
        end
  end
  /* 3° Output τ as τ_{pO}. */
    τ_{pO} ← τ;
end
```

τ_{p_O} is available by the above algorithm. Therefore, $\gamma_{migrate}$ can be obtained from the formula $\gamma_{migrate} = (\tau_{p_O} - \tau_{p_I}) - \Delta\tau_{N_{new}}$. Here, $\Delta\tau_{N_{new}} = \Sigma_{t_i^n \in \rho^n} d_i^n$, and ρ^n is the longest path of N_{old} from p_I to p_O with delay time d_i^n as its weight. The time complexity of this algorithm is $O((\tau_{p_O} - \tau_{cs}) \mid P^n \parallel T^n \mid)$.

5. Performance Evaluation

In our evaluation experiment, we generate the new WF-nets by applying change-operations proposed in section 3 to the prepared old WF-nets. We apply randomly selected change- operations to an old WF-net for 1 ~ 5 times. For each changed new WF-net, we transfer cases from an old WF-net to the new WF-net according to the method proposed in [5], and compute change time of Migrate and Ellis et al.'s three change types by the computing method proposed in section 4. In the following, we first discuss the effect of individual change- operations, then investigate the change time of 270 examples respectively for each change type and finally compare Migrate with Ellis et al.'s three change types

Table 1. Computation results of $\gamma_{migrate}$ when applying a change-operation.

	10	11	12	13	14	15	16	17	18	19	20	Ave.
Op.1	79.5	63.8	50.8	42.7	36.7	31.4	26.8	22.7	19.0	15.7	12.7	36.5
Op.2	95.2	79.5	66.3	55.0	45.4	38.5	33.8	29.7	26.0	22.7	20.2	46.6
Op.3	11.6	3.6	1.6	0.8	0.4	0.2	0.1	0.1	0.0	0.0	0.0	1.7
Op.4	48.4	33.3	25.2	20.9	17.5	14.6	12.0	9.8	7.8	5.9	4.3	18.2
Op.5	84.2	70.7	59.4	49.7	41.4	34.1	27.7	21.9	16.8	12.1	8.4	38.8
Op.6	8.1	3.4	1.6	0.8	0.4	0.2	0.1	0.1	0.0	0.0	0.0	1.3
Op.7	17.6	9.0	4.4	2.6	1.3	0.5	0.3	0.1	0.0	0.0	0.0	3.3
Op.8	7.7	3.4	1.6	0.8	0.4	0.2	0.1	0.1	0.0	0.0	0.0	1.3
Op.9	7.9	3.4	1.6	0.8	0.4	0.2	0.1	0.1	0.0	0.0	0.0	1.3
Op.10	9.9	5.2	3.0	1.5	0.8	0.5	0.2	0.1	0.0	0.0	0.0	1.9
Ave.	37.0	27.5	21.6	17.6	14.5	12.0	10.1	8.5	7.0	5.6	4.6	15.1

5.1 Experimental result on individual operation

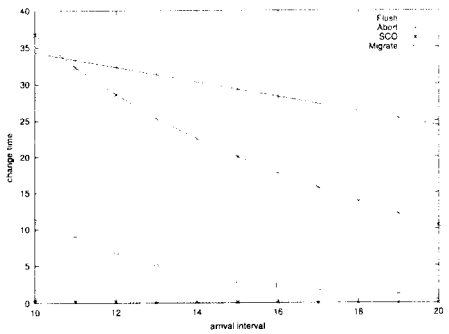
We first generated 9 old WF-nets $N_{old_1}, \cdots, N_{old_9}$ where N_{old_1}, N_{old_2}, N_{old_3} have 10 transitions ($|T^o|=10$), $N_{old_4}, N_{old_5}, N_{old_6}$ have 30 transitions ($|T^o|=30$), and N_{old_7}, N_{old_8}, N_{old_9} have 50 transitions ($|T^o|=50$). Then, in order to survey how individual change-operations have an effect on change time in Migrate, we generated 10 new WF-nets by applying an only change-operation to each old WF-net. Namely, we generated totally 90 ($=9\times10$) new WF-nets $N_{new_1}, \cdots, N_{new_{90}}$. We increased arrival interval d^* of input tokens from 10 to 20, and computed $\gamma_{migrate}$ of these examples for each d^*. As a result, we show the average of $\gamma_{migrate}$ to examples of $|T^o|=50$ in Table 1. Here, the first row and the first column express the arrival interval d^* of input token and the proposed change-operations, respectively.

From Table 1, it is clear that $\gamma_{migrate}$ in "Sequential insertion of activity", "Parallel insertion of activity", "Replacement of activity" and "Sequential exchange of activities" are longer than any other change-operations. Because, in a new WF-net applying such change-operations, a process must be redone in front of the activity not completed in N_{old}. Therefore, change times related to such change-operations tend to be long. Other change- operations on the contrary have very short change times.

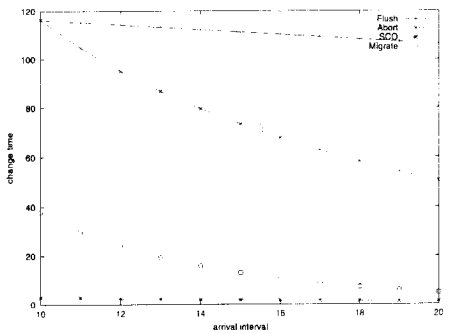
5.2 Results and Discussion about Migrate and Other Change Types

From the above experimental results, it is clear that there is a large difference on change time of Migrate between change-operations. Therefore, to take the average, we applied randomly selected change-operations to an old WF-net and obtained 270 examples. Concretely, for each old WF-net

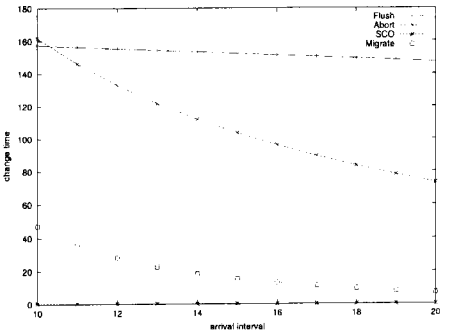
mentioned above, we generated 30 new WF-nets by applying change-operations randomly $1 \sim 5$ times, i.e. we generated totally 270 $(=9\times30)$ new WF-nets N_{new_1} , $\cdots$, $N_{new_{270}}$. Then we computed change time of each change type to all the examples and compared Migrate with Ellis et al.'s three change types. As a result, we show the comparison results on change time for the four change types in Fig.13.



(a) $|T^o|=10$.



(b) $|T^o|=30$.



(c) $|T^o|=50$.

Figure 13. Comparison on change time of Migrate and Ellis's three change types.

We found from Fig.13 that change times of any change types decrease with the increase of d^*. However, the degree of slope changes with change types. The change time of Flush decreases almost linearly. On the other hand the change time of Abort and Migrate decrease nonlinearly and the slope is large when d^* becomes short. This is related to, according to which WF-net (either N_{old} or N_{new}) the cases in N_{old} are handled after dynamic changed. In fact, these cases are handled according to N_{old} in Flush and SCO, but are handled according to N_{new} in Abort and Migrate.

Moreover, we found that Migrate has shorter change times than Flush and Abort, and SCO has shorter change times than Migrate. The difference of change times of Migrate and Abort depends on the number of activities not redone. In SCO, cases in N_{new} cannot be completed and kept waiting in front of the output place p_o until all cases in N_{old} are completed, in order to keep the order of FIFO. Therefore the change time of SCO takes a small value, unless the whole execution time is greatly improved by the change.

6. Concluding Remarks

In this paper, we have done the performance evaluation of Migrate on change time. We have firstly proposed a method of generating changed workflow nets. Concretely, we have proposed a set of basic operations to change workflow nets, called change-operations, and show the way to apply them to generate new workflow nets from old workflow nets. Then we have presented a method of computing change time of Migrate based on a previously proposed Petri net model. Finally we have done experiment of computer simulation for 270 examples generated by our change-operations. Our experiment shows that

(i) Migrate is better than Flush and Abort.
(ii) SCO is better than Migrate.

SCO is the best on change time among the four change types. However, SCO has a problem of conflict of resource, because both N_{old} and N_{new} exist for some time after the dynamic change is required. Moreover, it may be impossible to use both old workflow and new workflow together depending on business and

thus in such cases SCO cannot be applied. In this meaning, Migrate is not so bad compared with SCO. Therefore Migrate is a reasonably good type for dynamic changes.

As a future work, we will do the performance evaluation of Migrate on another evaluation measure, transient time [8], and compare Migrate with Ellis et al.'s three change types.

References

[1] Lawrence, P. ed., WfMC Workflow Handbook 1997, John Wiley & Sons, 1997.

[2] Aalst, W.M.P.V.D., Exterminating the dynamic change bug-A concrete approach to support workflow change, BETA Working Paper Series, WP 51, Eindhoven University of Technology, 2000.

[3] Ellis, C., Keddara, K. and Rozenberg, G., Dynamic Change Within Workflow Systems, Proc. ACM Conference on Organizational Computing Systems, pp.10-21, 1995.

[4] Sadiq, S.W., Marjanovic, O. and Orlowska, N.E., Managing Change and Time in Dynamic Workflow Processes, Int. J. Cooperative Information Systems, vol.9, no.1&2, pp.93-116, 2000.

[5] Yamaguchi, S., Mishima, A., Ge, Q.W. and Tanaka, M., Modeling and Implementation of Migrate Dynamic Workflow Changes, IEICE Trans. Fundamentals, vol.E86-A, no.6, 2003 (to appear).

[6] Liu, C., Orlowska, M.E. and Li, H., Automating handover in dynamic workflow environments, Proc. 10th Int. Conf. Advances in Inf. System. Eng. (CAiSE98), pp.139-157, 1998.

[7] Yamaguchi, S., Ge, Q.W. and Tanaka, M., Performance Evaluation on Change Time of Dynamic Workflow Changes, IEICE Trans. Fundamentals, vol.E83-A, no.11, pp.2177-2187, 2000.

[8] Yamaguchi, S., Shiode, Y., Ge, Q.W. and Tanaka, M., Performance evaluation on transient time of dynamic workflow changes, IEICE Trans. Fundamentals, vol.E84-A, no.11, pp. 2852-2864, 2001.

[9] Aalst, W.M.P.V.D., The Application of Petri Nets to Workflow Management, J. Circuits, Systems, Computers, vol.8, no.1, pp.21-65, 1998.

[10] The Institute of Electrical Engineers of Japan, Practice of Workflow, JUSE Press, pp.78-89, 1999 (in Japanese).