Mining workflow processes from distributed workflow enactment event logs

Kwanghoon Pio Kim*

Collaboration Technology Research Lab
Department of Computer Science
Kyonggi University, South Korea
E-mail: kwang@kgu.ac.kr

*Corresponding author

Abstract: Workflow management systems help to execute, monitor and manage work process flow and execution. These systems, as they are executing, keep a record of who does what and when (e.g. log of events). The activity of using computer software to examine these records, and deriving various structural data results is called workflow mining. The workflow mining activity, in general, needs to encompass behavioral (process/control-flow), social, informational (data-flow), and organizational perspectives; as well as other perspectives, because workflow systems are "people systems" that must be designed, deployed, and understood within their social and organizational contexts. This paper particularly focuses on mining the behavioral aspect of workflows from XML-based workflow enactment event logs, which are vertically (semantic-driven distribution) or horizontally (syntactic-driven distribution) distributed over the networked workflow enactment components. That is, this paper proposes distributed workflow mining approaches that are able to rediscover ICN-based structured workflow process models through incrementally amalgamating a series of vertically or horizontally fragmented temporal workcases. And each of the approaches consists of a temporal fragment discovery algorithm, which is able to discover a set of temporal fragment models from the fragmented workflow enactment event logs, and a workflow process mining algorithm which rediscovers a structured workflow process model from the discovered temporal fragment models. Where, the temporal fragment model represents the concrete model of the XML-based distributed workflow fragment events log.

Keywords: Distributed workflow management system; Distributed events log; Distributed workflow process mining; Workflow fragmentation

Biographical notes: Kwanghoon Pio Kim is a full Professor of Computer Science Department and the founder and supervisor of the collaboration technology research laboratory at Kyonggi University, South Korea. Also, he is in charge of the director of the computerization and information institute in Kyonggi University, and was in charge of the director of the contents convergence software research center established at 2007 as a new GRRC project funded by the Gyeonggi Provincial Government, Republic of Korea. He received B.S. degree in computer science from Kyonggi University in 1984. And he received M.S. degree in computer science from Chungang University in 1986. He also received his M.S. and Ph.D. degree from the Computer Science Department of University of Colorado at Boulder, in 1994 and 1998, respectively. He had worked as researcher and developer at Aztek Engineering, American Educational Products Inc., and IBM in USA, as well as at Electronics

and Telecommunications Research Institute (ETRI) in South Korea. In present, he is a vice-chair of the BPM Korea Forum. He has been in charge of a country-chair (Korea) and ERC vice-chair of the Workflow Management Coalition. He has also been on the editorial board of the journal of KSII, and the committee member of the several conferences and workshops. His research interests include groupware, workflow systems, BPM, CSCW, collaboration theory, Grid/P2P distributed systems, process warehousing and mining, workflow-supported social networks and analysis, and process-aware information systems.

1. Introduction

A Workflow Management System (WfMS) is defined as a system that (partially) automates the definition, creation, execution, and management of work processes through the use of software that is able to interpret the process definition, interact with workflow participants, and invoke the use of IT tools and applications. Furthermore, the platforms for WfMSs' deployment and enactment have been swiftly evolving into the distributed computing environments, such as clustering, grid, P2P and cloud computing environments. Particularly, in the paper, as a platform, we consider the enterprise workflow grid (Kim, 2007) and the enterprise workflow cloud (Kim, 2007) computing environments. Note that the fragments of workflow models are disseminated over the workflow enactment nodes of the platform, and that their enactment event logs formatted in XML are recorded onto themselves.

Such distributed workflow management systems are becoming a catalyst for triggering emergence of the concept of distributed workflow mining that rediscovers several perspectives—control flow, data flow, social, and organizational perspectives—of workflows from the scattered workflow execution event histories (logs) collected at runtime of distributed workflow models fragmented from an original workflow model. In this paper, we particularly focus on mining the behavioral—control flow—perspective (Park & Kim, 2008) of the fragmented workflow models. In general, a workflow model is described by several entities, such as activity, role, actor, invoked applications, and relevant data, and where, steps of a work process are called activities (jobs or transactions) that flow through the system are called workcases (Ellis, 1979) or workflow instances. The control flow perspective, which we particularly call a workflow process, specifies the transition precedence—sequential, conjunctive(AND) and disjunctive(OR) execution sequences—among the activities, and it is represented by the concept of workflow process model defined in this paper by using the graphical and formal notations of the information control net (ICN) (Kim & Ellis, 2007). Also, we assume that the workflow process model keeps the proper nesting and the matched pairing properties in modeling the conjunctive and the disjunctive transitions—AND-split, AND-join nodes and OR-split, OR-join nodes, which are the basic properties of a structured workflow process model (Liu & Kumar, 2005; Kim & Ellis, 2007).

Based upon the concept of the structured workflow process model (Liu & Kumar, 2005), we propose a series of distributed workflow process mining approaches that play a theoretical basis for implementing a distributed workflow mining system that is able to rediscover structured workflow process models from a series of fragmented XML-based workflow enactment event logs (Kim, 2006), which are horizontally (instance-based distribution) or/and vertically (activity-based distribution) distributed over the networked workflow enactment components. Each of the fragmented workflow event logs is typically an interleaved list of events from numerous workcases—workflow instances—allocated to the corresponding workflow enactment component. By examining and combining the fragmented logs, we can discover the temporal ordering of activity executions for each workcase, which is dubbed a temporal workcase, and then infer a general structured workflow process structure by amalgamating the discovered temporal workcases.

As a simple example, suppose we examine the fragmented logs of a workflow process that has four activities, α_1, α_2, α_3, and α_4, each of which is vertically/horizontally fragmented and allocated into a different workflow enactment component deployed over a distributed computing environment. Suppose also that all four activities are always executed in some order by each workcase, even though the enactments of the activities are conducted in different workflow enactment components. If we observe over a large number of workcases that α_1 is always executed first and α_4 is always executed last, then we can begin to piece together a workflow process model that requires α_1 to complete before all other activities, and α_4 to execute after all others. If we find workcases in the log where α_2 begins before α_3, and other cases where α_2 begins after α_3, then we can infer that the workflow process begins with α_1, after it completes, α_2 and α_3 execute concurrently (Conjunctive Transition: AND Control Flow); and after they both complete, then α_4 executes.

This is an extremely simplified example that ignores the other important control transition construct—Disjunctive Transition (OR Control Flow)—and their combinations. However, it is enough to explain the basic principle of the distributed workflow process mining. So, in the remainder of this paper, we are going to show that our distributed workflow mining approaches are able to handle all of the possible activity execution cases through the concepts of fragmented temporal workcases. At first, the next section presents the meta-model of the structured workflow process model with graphical and formal notations, and describes how to fragment the model through vertical, horizontal or hybrid fragmentation approach. In the main sections of this paper, we firstly describe an XML-based workflow enactment event log format, and illustrate distributed workflow process mining approaches and the detailed descriptions of the temporal workcase discovery algorithms and the workflow process mining algorithms with some examples. Finally, we discuss the constraints of the proposed approaches and algorithms and the related work.

2. Workflow fragmentation

This paper basically assumes that the information control net methodology (Ellis, 1979) is used to represent workflow process models. The information control net (ICN) was originally developed to describe and analyze information flow by capturing several entities within office procedures, such as activities, roles, actors, activity precedence, applications, and repositories. It has been used within actual as well as hypothetical automated offices (1) to yield a comprehensive description of activities, (2) to test the underlying office description for certain flaws and inconsistencies, (3) to quantify certain aspects of office information flow, and (4) to suggest possible office restructuring permutations. Especially, we define the structured workflow process model (Liu & Kumar, 2005) preserving the proper nesting and matched pairing properties. Once a structured workflow process is defined, it needs to be fragmented for being enacted over the distributed workflow enactment platform, like enterprise workflow grid (Kim, 2007). So, this section describes the fragmentation methods (Kim, 2012)—vertical, horizontal and hybrid fragmentation—of workflow processes defined by the structured workflow process model.

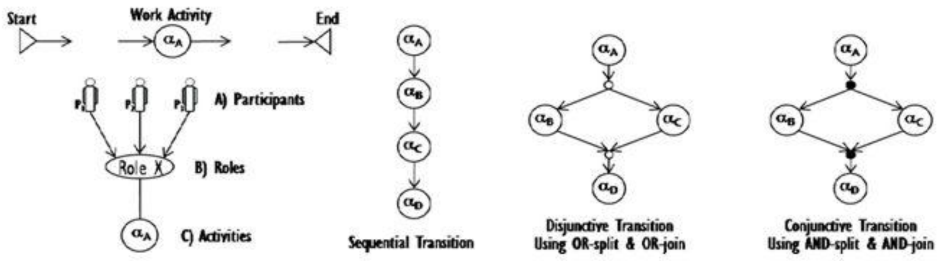

Start
Work Activity
End
A) Participants
Role X
B) Roles
C) Activities
Sequential Transition
Disjunctive Transition Using OR-split & OR-join
Conjunctive Transition Using AND-split & AND-join

Fig. 1. Graphical notations

2.1. *ICN-based structured workflow process model*

We focus on the activities and their related information flows by defining the ICN-based structured workflow process model, which is the target workflow model of the distributed workflow process mining approach proposed in the paper, through a set of graphical constructs and their formal representation.

2.1.1. *Graphical representation*

As shown in Fig. 1, a structured workflow process model consists of a set of activities connected by temporal orderings called activity transitions. In other word, it is a predefined set of work steps, called activities, with a partial ordering (or control flow) by combining sequential transition types, disjunctive transition types (after activity α_A, do activity α_B or α_C, alternatively) with predicates attached, and conjunctive transition types (after activity α_A , do activities α_B and α_C concurrently). Particularly, the disjunctive and conjunctive transition types must keep the structured properties of proper nesting and matched pairing in defining workflow process models. An activity may be either a compound activity containing another sub-process, or a basic unit of work, which is called a work activity. The work activity is executed in one of three modes: manual, automatic, or hybrid, and is mapped to a role that takes charge of enacting the corresponding one, as shown in the left-hand side of Fig. 1. Fig. 2 is a simple example of the structured workflow process model with three roles and five participants. Note that the AND-Control nodes (AND-split and AND-join that are presented by solid dots(•)), and the OR-Control nodes (OR-split and OR-join that are represented by hollow dots(◦)), in a model must be properly nested and matched paired in order to build a structured workflow process model (Ellis, Kim, & Rembert, 2006; Ellis, Rembert, Kim, & Wainer, 2006; Kim & Ellis, 2007).

2.1.2. *Formal representation*

The structured workflow process model needs to be represented by a formal notation that provides a means to eventually specify the model in textual language or in database, and both. The following definition is the formal representation of the structured workflow process model:

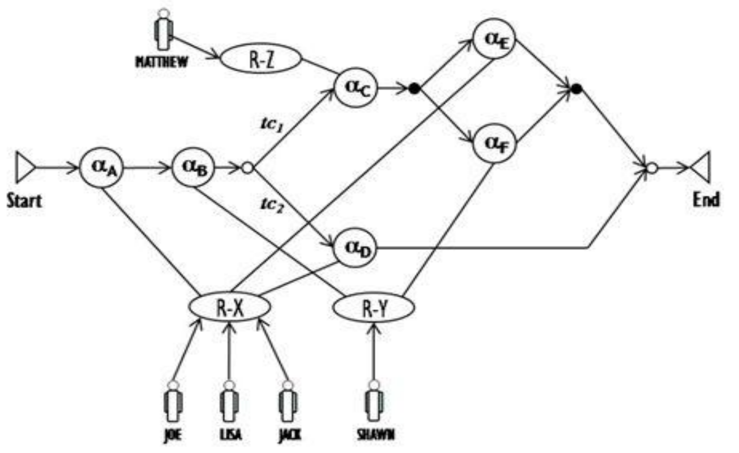


Fig. 2. A simple structured workflow process model

Definition 1. Structured Workflow Process Model (SWPM). A basic structured workflow process model is formally defined through 4-tuple $I' = (\delta, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$ over an activity set **A**, a role set **R**, a participant set **P**, and a transition condition set **T**, where

- **I** is a finite set of initial input repositories, assumed to be loaded with information by some external process before execution of the model;
- **O** is a finite set of final output repositories, which is containing information used by some external process after execution of the model;
- $\delta = \delta_i \cup \delta_o$,
  where, $\delta_o : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a multi-valued mapping function of an activity to its set of (immediate) successors, and $\delta_i : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a multi-valued mapping function of an activity to its set of (immediate) predecessors;
- $\varepsilon = \varepsilon_a \cup \varepsilon_r$,
  where, $\varepsilon_r : \mathbf{A} \longrightarrow \wp(\mathbf{R})$ is a single-valued function mapping an activity to a role, and $\varepsilon_a : \mathbf{R} \longrightarrow \wp(\mathbf{A})$ is a multi-valued function mapping a role to its sets of associated activities;
- $\pi = \pi_r \cup \pi_p$,
  where, $\pi_p : \mathbf{R} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a role to its sets of associated participants (actors), and $\pi_r : \mathbf{P} \longrightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a participant to its sets of associated roles;
- $\kappa = \kappa_i \cup \kappa_o$,
  where, $\kappa_i : \mathbf{A} \longrightarrow \wp(\mathbf{T})$ is a multi-valued mapping function of an activity to its incoming transition-conditions ($\in \mathbf{T}$) on each arc, $(\delta_i(\alpha), \alpha)$, and $\kappa_o : \mathbf{A} \longrightarrow \wp(\mathbf{T})$ is a multi-valued mapping function of an activity to its outgoing transition-conditions ($\in \mathbf{T}$) on each arc, $(\alpha, \delta_o(\alpha))$;

Table 1
Formal representation of the structured workflow process model

$I' = (\delta, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$ over **A**, **R**, **P**, **T**	**/* The Structured Workflow Process Model**
A = $\{\alpha_{start}, a_A, \alpha_B, \alpha_C, \alpha_D, \alpha_E, \alpha_F, \alpha_{end}\}$ **/* Activities**	
R = $\{\eta_{start}, \eta_{R-X}, \eta_{R-Y}, \eta_{R-Z}, \eta_{end}\}$ **/* Roles**	
P = $\{o_{start}, o_{jack}, o_{joe}, o_{lisa}, o_{matthew}, o_{shawn}, o_{end}\}$ **/* Participants**	
T = $\{d(default), tc_1, tc_2\}$ **/* Transition Conditions**	
I = $\emptyset$ **/* Initial Input Repositories**	
O = $\emptyset$ **/* Final Output Repositories**	
$\delta_i(\alpha_{start}) = \emptyset;$	$\delta_o(\alpha_{start}) = \{\{\alpha_A\}\};$
$\delta_i(\alpha_A) = \{\{\alpha_{start}\}\};$	$\delta_o(\alpha_A) = \{\{\alpha_B\}\};$
$\delta_i(\alpha_B) = \{\{\alpha_A\}\};$	$\delta_o(\alpha_B) = \{\{\alpha_C\},\{\alpha_D\}\};$
$\delta_i(\alpha_C) = \{\{\alpha_B\}\};$	$\delta_o(\alpha_C) = \{\{\alpha_E, \alpha_F\}\};$
$\delta_i(\alpha_D) = \{\{\alpha_B\}\};$	$\delta_o(\alpha_D) = \{\{\alpha_{end}\}\};$
$\delta_i(\alpha_E) = \{\{\alpha_C\}\};$	$\delta_o(\alpha_E) = \{\{\alpha_{end}\}\};$
$\delta_i(\alpha_F) = \{\{\alpha_C\}\};$	$\delta_o(\alpha_F) = \{\{\alpha_{end}\}\};$
$\delta_i(\alpha_{end}) = \{\{\alpha_D\},\{\alpha_E, \alpha_F\}\};$	$\delta_o(\alpha_{end}) = \emptyset;$
$\delta = \delta_i \cup \delta_o$	
$\varepsilon_r(\alpha_{start}) = \{\eta_{start}\};$	$\varepsilon_a(\eta_{start}) = \{\alpha_{start}\};$
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-X}\};$	$\varepsilon_a(\eta_{R-X}) = \{\alpha_A, \alpha_D, \alpha_E\};$
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-Y}\};$	$\varepsilon_a(\eta_{R-Y}) = \{\alpha_B, \alpha_F\};$
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-Z}\};$	$\varepsilon_a(\eta_{R-Z}) = \{\alpha_C\};$
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-X}\};$	$\varepsilon_a(\eta_{end}) = \{\alpha_{end}\};$
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-X}\};$	
$\varepsilon_r(\alpha_{start}) = \{\eta_{R-Y}\};$	
$\varepsilon_r(\alpha_{start}) = \{\eta_{end}\};$	
$\varepsilon = \varepsilon_a \cup \varepsilon_r$	
$\pi_p(\eta_{start}) = \{o_{start}\};$	$\pi_r(o_{start}) = \{\eta_{start}\};$
$\pi_p(\eta_{R-X}) = \{o_{jack}, o_{joe}, o_{lisa}\};$	$\pi_r(o_{jack}) = \{\eta_{R-X}\};$
$\pi_p(\eta_{R-Y}) = \{o_{shawn}\};$	$\pi_r(o_{joe}) = \{\eta_{R-X}\};$
$\pi_p(\eta_{R-Z}) = \{o_{matthew}\};$	$\pi_r(o_{lisa}) = \{\eta_{R-X}\};$
$\pi_p(\eta_{end}) = \{o_{end}\};$	$\pi_r(o_{shawn}) = \{\eta_{R-Y}\};$
	$\pi_r(o_{matthew}) = \{\eta_{R-Z}\};$
	$\pi_r(o_{end}) = \{\eta_{end}\};$
$\pi = \pi_r \cup \pi_p$	
$\kappa_i(\alpha_{start}) = \emptyset;$	$\kappa_o(\alpha_{start}) = \{d\};$
$\kappa_i(\alpha_A) = \{d\};$	$\kappa_o(\alpha_A) = \{d\};$
$\kappa_i(\alpha_B) = \{d\};$	$\kappa_o(\alpha_B) = \{\{tc_1\}, \{tc_2\}\};$
$\kappa_i(\alpha_C) = \{tc_1\};$	$\kappa_o(\alpha_C) = \{d\};$
$\kappa_i(\alpha_D) = \{tc_2\};$	$\kappa_o(\alpha_D) = \{d\};$
$\kappa_i(\alpha_E) = \{d\};$	$\kappa_o(\alpha_E) = \{d\};$
$\kappa_i(\alpha_F) = \{d\};$	$\kappa_o(\alpha_F) = \{d\};$
$\kappa_i(\alpha_{end}) = \{d\};$	$\kappa_o(\alpha_{end}) = \emptyset;$
$\kappa = \kappa_i \cup \kappa_o$	

2.1.3. Starting and terminating nodes

Additionally, the execution of a workflow process model commences by a single λ transition-condition. So, we always assume without loss of generality that there is a single starting node (α_I)t the commencement, it is assumed that all input repositories in the set ***I*** have been initialized with data by the external system:

$$\exists \alpha_I \in \mathrm{A} \mid \delta_i(\alpha_I) = \emptyset \wedge \kappa_o(\alpha_I) = \{\{\lambda\}\}.$$

The execution is terminated with any one λ output transition-condition. Also we assume without loss of generality that there is a single terminating node (α_F). The set of output repositories ***O*** is data holders that may be used after termination by the external system:

$$\exists \alpha_F \in \mathrm{A} \mid \delta_o(\alpha_F) = \emptyset \wedge \kappa_i(\alpha_F) = \{\{\lambda\}\}.$$

2.1.4. Implication: Structured modeling methodology preserving the proper nesting and the matched pairing properties

Given a formal definition, the structured ordering of a workflow process model can be interpreted as follows: For any activity α ($\delta = \delta_i \cup \delta_o$), in general,

$$\begin{aligned}\delta_o(\alpha) = \{ & \\ & \{\beta_{11}, \beta_{12}, \ldots, \beta_{1m(1)}\}, \\ & \{\beta_{21}, \beta_{22}, \ldots, \beta_{2m(2)}\}, \\ & \ldots, \\ & \{\beta_{n1}, \beta_{n2}, \ldots, \beta_{nm(n)}\} \\ & \}\end{aligned}$$

means that upon completion of activity α, either a set of transitions that simultaneously initiates all of the activities β_{i1} through $\beta_{im(i)}$ occurs, or a transition that only one value of β_{i1} $i(1 \leq i \leq n)$ is selected as the result of a decision made within activity α occurs, or both. In general, if $n = 1$, then no decision is needed and α is not a decision node. If also $m(i) = 1$ for all i , then no parallel processing is initiated by completion of α. (Note that $\beta_{ij} \in \{\forall \alpha, \{\emptyset\}\}$, $(1 \leq i \leq n)$, $(1 \leq j \leq m)$). In the SWPM graphical notation, the former, that an activity has a conjunctive (or parallel) outgoing transition, is represented by a solid dot—AND-split, and the latter, that an activity has a disjunctive (or decision) outgoing transition, is represented by a hollow dots—OR-split. And also,

$$\begin{aligned}\delta i(\alpha) = \{ & \\ & \{\beta_{11}, \beta_{12}, \ldots, \beta_{1m(1)}\}, \\ & \{\beta_{21}, \beta_{22}, \ldots, \beta_{2m(2)}\}, \\ & \ldots, \\ & \{\beta_{n1}, \beta_{n2}, \ldots, \beta_{nm(n)}\} \\ & \}\end{aligned}$$

means that upon commencement of activity α, either all the activities, β_{i1} through $\beta_{im(i)}$, simultaneously completes, or only one transition β_{i1} out of the activities β_{11} through β_{n1}, $i(1 \leq i \leq n)$ completes, or both. In general, if $m(i) = 1$ for all i, then no parallel processing is completed before the commencement of α. In the SWPM graphical notation, the former, that an activity has a conjunctive (or parallel) incoming transition, is represented by a solid dot—AND-join, and the latter, that an activity has a disjunctive (or decision) incoming transition, is represented by a hollow dot—OR-join. Summarily, the following is to formally specify the basic transition types depicted in Fig. 1. Also, Table 1 is to represent the formal description of the structured workflow process model in Fig. 2.

(1) Sequential Transition
incoming $\rightarrow \delta_i(\alpha_B) = \{\{\alpha_A\}\}$; *outgoing* $\rightarrow \delta_o(\alpha_B) = \{\{\alpha_C\}\}$;

(2) OR Transition
or-split $\rightarrow \delta_o(\alpha_A) = \{\{\alpha_B\}, \{\alpha_C\}\}$; *or-join* $\rightarrow \delta_i(\alpha_D) = \{\{\alpha_B\}, \{\alpha_C\}\}$;

(3) AND Transition

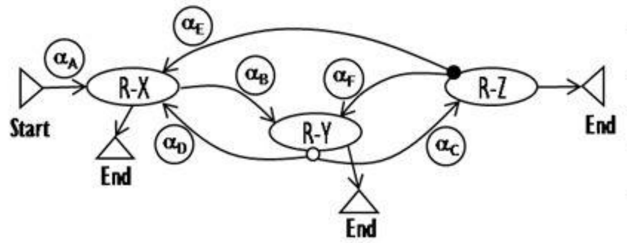
and-split $\rightarrow \delta_o(\alpha_A) = \{\{\alpha_B, \alpha_C\}\}$; *and-join* $\rightarrow \delta_i(\alpha_D) = \{\{\alpha_B, \alpha_C\}\}$;

2.2. Workflow fragmentation methods

Based upon the ICN-based structured workflow process model described in the previous subsection, this subsection defines the basic concept of workflow fragmentation (Kim, 2012). Conceptually speaking, the primary goal of the workflow fragmentation is to reasonably break a workflow process into fragments, and to distribute the fragments over the networked workflow engine's components running on an enterprise workflow grid computing environment (Kim, 2007). According to enacting the instances of the workflow process, each of the workflow engine components (that are associated to the enactment of the workflow process's instances) records its execution events into the corresponding local log. The local event logs are formatted in the XML-based fragmented workflow event log format extended from the original workflow enactment event logging mechanism (Park & Kim, 2010) and the language (Kim, 2006). From those XML-based distributed workflow event logs, it is possible to rediscover a structured workflow process by applying the distributed workflow process mining approaches to be proposed in the paper. At this moment, it is important to figure out how to fragment a workflow process, which can be done vertically, horizontally or in hybrid. The vertical fragmentation implies semantic-driven distribution purporting the collaborative enactment of the fragments, while the horizontal fragmentation works for syntactic-driven distribution mainly focusing on the instance types of the corresponding workflow process. And the hybrid fragmentation implies applying the vertical fragmentation approach to each of the fragments broken from the horizontally fragmentation approach. In this section, we describe the basic principles of the vertical and horizontal fragmentation methods, and the details of them.

2.2.1. Vertical fragmentation

A structured workflow process model consists of a set of activities and their temporal precedences. In order to enact the model on a distributed computing environment (which is supposed to be an enterprise grid computing environment (Kim, 2007)), it is necessary to break the model into fragments and distribute them over the computing nodes. Actually, the meaning of the vertical fragmentation implies semantic grouping of the activities of the model, and each group can be allocated into each node of the computing environment. Of course the vertical fragmentation can be done by random grouping method, and it, however, ought not to be a reasonable approach, because it's hard to estimate its operational performance, as we know.



Node	Activity Fragments
R-X	α_A α_D α_E
R-Y	α_B α_F
R-Z	α_C

Fig. 3. The role-based vertical fragmentation result

Conclusively, the vertical fragmentation based on the semantic grouping method is to make activity-groups based upon the semantic components—roles and actors—assigned to the structured workflow process model. As an example, we present one of the semantic grouping methods, which we dub it the role-based workflow fragmentation approach (Kim, 2012) that is made up of the role-based workflow fragment model and its automatic generation algorithm. The fundamental idea of the approach is that the activities to be performed by a same role are distributed to a same computing node. We apply the approach to the structured workflow process model presented in the previous subsection, and its vertical fragmentation result is illustrated in Fig. 3. The left-hand side of the figure is the graphical representation of the role-based workflow fragment model, and the right-hand side is the final activity fragments and the distribution status to the associated computing nodes.

The formal definition of the role-based workflow fragment model is described in [*Definition 2*], and its graphical primitives are oval(node), directed arc with label(activity), solid dot(•: parallel) and hollow dot(◦: decision) as shown in Fig. 3. The model represents two types of information—node flows and fragmented activities through which we are able to get precedence (predecessor/successor) relationships among nodes as well as distributed activities of each node. For an instance, the activities, α_A, α_D , α_E , on the incoming directed arcs of the node, η_{R-X}, are the assigned activities to the corresponding node.

Definition 2. Role-based Workflow Fragment Model. A role-based workflow fragment model is formally defined as $\mathfrak{R} = (\xi, \vartheta, \mathbf{S}, \mathbf{E})$, over a set **R** of roles and a set **A** of activities, where,

- **S** is a finite set of the initial nodes;
- **E** is a finite set of the final nodes;
- $\xi = \xi_i \cup \xi_o$ /* Node Flow: successors and predecessors */
  where, $\xi_o : \mathbf{R} \rightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a node to its sets of (immediate) successors, and $\xi_i : \mathbf{R} \rightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a node to its sets of (immediate) predecessors;
- $\vartheta = \vartheta_i \cup \vartheta_o$ /* Fragments and Neighbor Fragments */
  where, $\vartheta_i : \mathbf{A} \rightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a set of fragmented activities into the node, η; and $\vartheta_o : \mathbf{A} \rightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a set of neighbor fragments' activities to the node, η;

In terms of fragmenting of a workflow process, it is definitely necessary to automatically construct a role-based workflow fragment model. In other words, it is very important to provide an automatic methodology for implementing the semantic grouping method. Therefore, we conceive an algorithm for automatically construct the role-based workflow fragment model from an ICN-based workflow model. The following is the algorithm that is called the role-based workflow fragmentation algorithm. The time complexity of the vertical fragmentation algorithm is $O(n)$, where n is the number of activities in the structured workflow process model, because the function has a single for-loop with repeating as many as the number of activities. Therefore, the overall time complexity is $O(n)$.

```
PROCEDURE Role-based Workflow Fragmentation Algorithm
Input A Structured Information Control Model, $\Gamma = (\delta, \rho, \lambda, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$;
Output A Role-based Workflow Fragmentation Model, $\Re = (\xi, \vartheta, \mathbf{S}, \mathbf{E})$;
BEGIN
  FOR $(\forall\alpha \in \mathbf{A})$ DO
          /* $\xi = \xi_i \cup \xi_o$ */
      Add $\varepsilon_r(\alpha)$ To $\xi_i(\varepsilon_r(\textit{all members of } \delta_o(\alpha)))$;
      Add $\varepsilon_r(\textit{all members of } \delta_o(\alpha))$ To $\xi_o(\varepsilon_r(\alpha))$;
          /* $\vartheta = \vartheta_i \cup \vartheta_o$ */
      Add $\alpha$ To $\vartheta_i(\varepsilon_r(\alpha))$;
      Add $\delta_o(\alpha)$ To $\vartheta_o(\varepsilon_r(\alpha))$;
  END-FOR
END-PROCEDURE
```

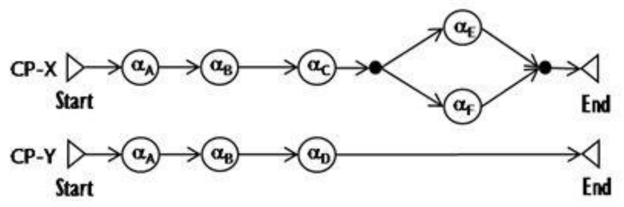
Table 2
The result of the role-based workflow fragmentation algorithm

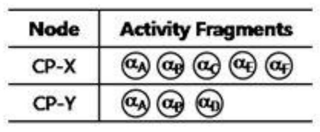
$\Re = (\xi, \vartheta, \mathbf{S}, \mathbf{E})$ over **A**, **R** /* **The Role-based Workflow Fragmentation Model**	
A = $\{\alpha_{start}, a_A, \alpha_B, \alpha_C, \alpha_D, \alpha_E, \alpha_F, \alpha_{end}\}$ /* **Activities**	
R = $\{\eta_{start}, \eta_{R-X}, \eta_{R-Y}, \eta_{R-Z}, \eta_{end}\}$ /* **Roles**	
S = $\emptyset$ /* **Initial Nodes**	
E = $\emptyset$ /* **Final Nodes**	
ξ_i: **Predecessors**	ξ_o: **Successors**
$\xi_i(\eta_{start}) = \emptyset;$	$\xi_o(\eta_{start}) = \{\{\eta_{R-X}\}\};$
$\xi_i(\eta_{R-X}) = \{\{\eta_{start}\}, \{\eta_{R-Y}\}, \{\eta_{R-Z}\}\};$	$\xi_o(\eta_{R-X}) = \{\{\eta_{R-X}\}, \{\eta_{end}\}\};$
$\xi_i(\eta_{R-Y}) = \{\{\eta_{R-X}\}, \{\eta_{R-Z}\}\};$	$\xi_o(\eta_{R-Y}) = \{\{\{\eta_{R-X}\}, \{\eta_{R-Z}\}\}, \{\eta_{end}\}\};$
$\xi_i(\eta_{R-Z}) = \{\{\eta_{R-Y}\}\};$	$\xi_o(\eta_{R-Z}) = \{\{\{\eta_{R-X}, \eta_{R-Y}\}\}, \{\eta_{end}\}\};$
$\xi_i(\eta_{end}) = \{\{\eta_{R-X}\}, \{\eta_{R-Y}\}, \{\eta_{R-Z}\}\};$	$\xi_o(\eta_{end}) = \emptyset;$
$\xi = \xi_i \cup \xi_o$	
ϑ_i: **Fragments**	ϑ_o: **Neighbor Fragments**
$\vartheta_i(\eta_{start}) = \{\{\alpha_{start}\}\};$	$\vartheta_o(\eta_{start}) = \{\{\alpha_A\}\};$
$\vartheta_i(\eta_{R-X}) = \{\{\alpha_A\}, \{\alpha_D\}, \{\alpha_E\}\};$	$\vartheta_o(\eta_{R-X}) = \{\{\alpha_B\}, \{\alpha_{end}\}\};$
$\vartheta_i(\eta_{R-Y}) = \{\{\alpha_B\}, \{\alpha_F\}\};$	$\vartheta_o(\eta_{R-Y}) = \{\{\{\alpha_C\}, \{\alpha_D\}, \{\alpha_{end}\}\};$
$\vartheta_i(\eta_{R-Z}) = \{\{\alpha_C\}\};$	$\vartheta_o(\eta_{R-Z}) = \{\{\{\alpha_E, \alpha_F\}\}, \{\alpha_{end}\}\};$
$\vartheta_i(\eta_{end}) = \{\{\alpha_{end}\}\};$	$\vartheta_o(\eta_{end}) = \emptyset;$
$\vartheta = \vartheta_i \cup \vartheta_o$	

As result, we give the formal representation of the role-based workflow fragmentation model of the structured workflow process model in Table 2, which is automatically generated by applying the algorithm. As you can see, the table shows the node flow information and each node's fragmented activities based upon 3 nodes and 6 elementary activities.

2.2.2. Horizontal fragmentation

On the other hand, the conceptual meaning of horizontal fragmentation of a workflow process implies syntactical grouping of activities. That is, the syntactic components of the structured workflow process model, such as OR-nodes and AND-nodes, become the criteria for grouping the activities. Conclusively, the reachable control-paths (Kim & Ellis, 2006) of a structured workflow process become the horizontal fragments that are distributed into the computing nodes, as shown in Fig. 4. The left-hand side of the figure represents the reachable control pathes of the structured workflow process model introduced in the previous section, and the right-hand side shows the horizontal fragments, each of which can be distributed into one of the computing nodes, CP-X and CP-Y. As you see, in this horizontal fragmentation approach some activities like α_A, α_B may be duplicately grouped into different computing nodes.





Node	Activity Fragments
CP-X	α_A α_B α_C α_E α_F
CP-Y	α_A α_B α_D

Fig. 4. The horizontal fragmentation result

In order to formally define the horizontal fragmentation approach, it is necessary to define the controlpath-based fragment model (Kim, 2012) and its generation algorithm. The definition of the controlpath-based fragment model is given in [*Definition 3*], and the horizontal fragmentation algorithm described in the followings fragments a structured workflow process model, as an input, into several controlpath-based fragment models. The time complexity of the horizontal fragmentation algorithm is $O(n)$, where n is the number of activities in the structured workflow process model, because the function, H-FRAGMENTATION(), is recursively traversing each activity in only once. Therefore, the overall time complexity is $O(n)$.

Definition 3.Controlpath-based Fragment Model of a structured workflow process model. Let **W** be a **CpFN**, a control-path fragment net, that is formally defined as **CpFN** = (ϱ, κ, **I**, **O**) over a set of activities, $\mathbf{A}^{cp}$, and a set of transition-conditions, $\mathbf{T}^{cp}$, where

- $\varrho = \varrho_i \cup \varrho_o$
  where, ϱ_o: $\mathbf{A}^{cp} \longrightarrow \wp(\alpha \in \mathbf{A}^{cp})$ is a multi-valued mapping of an activity to its set of (immediate) successors, and ϱ_i : $\mathbf{A}^{cp} \longrightarrow \wp(\alpha \in \mathbf{A}^{cp})$ is a single-valued mapping of is a multi-valued mapping function of an activity to its set of (immediate) predecessors;
- $\beta = \beta_i \cup \beta_o$
  where, $\beta_i(\alpha)$: a set of control transition conditions, $\tau \in \mathbf{T}^{cp}$, on each arc, ($\beta_i(\alpha)$, α); and $\beta_o(\alpha)$: a set of control transition conditions, $\tau \in \mathbf{T}^{cp}$, on each arc, (α, β_o (α)), where $\alpha \in \mathbf{A}^{cp}$;
- **I** is a finite set of initial input repositories of the corresponding structured workflow process model;
- **O** is a finite set of final output repositories of the corresponding structured workflow process model;

```
PROCEDURE Controlpath-based Fragmentation Algorithm
Input A Structured Workflow Process Model, $\Gamma = (\delta, \gamma, \lambda, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$;
Output A Set of Controlpath-based Fragment Models (CpFNs), $\forall \mathbf{W} = (\varrho, \kappa, I, O)$;
Initialize CpN ← {∅};      /* The empty net of CpFN. */
PROCEDURE H-FRAGMENTATION(In s ← $\{\alpha_I\}$, CpFN) /* Recursive Function */
BEGIN
        $v \leftarrow s$; CpFN.$A^{cp} \leftarrow$ CpFN.$A^{cp} \cup \{ v \}$;
        WHILE (($u \leftarrow \delta_o(s)$;) = $\{\alpha_F\}$)
```

```
            SWITCH (What type of the activity, $u$, is?) DO
                Case 'serial-type activity':
                    $w \leftarrow u$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{w\}$;
                    CpFN.$\varrho_o(v) \leftarrow w$; CpFN.$\varrho_i(w) \leftarrow v$;
                    CpFN.$\beta_o(v) \leftarrow \kappa_o(s)$; CpFN.$\beta_i(v) \leftarrow \kappa_i(s)$;
                    break;
                Case 'conjunctive-type (AND-split) activity':
                    $w \leftarrow u$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{w\}$;
                    CpFN.$\varrho_o(v) \leftarrow w$; CpFN.$\varrho_i(w) \leftarrow v$;
                    CpFN.$\beta_o(v) \leftarrow \kappa_o(s)$; CpFN.$\beta_i(v) \leftarrow \kappa_i(s)$;
                    FOR (eachof $\forall\, \alpha \in \delta_o(u)$) DO
                        $x \leftarrow \alpha$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{x\}$;
                        CpFN.$\varrho_o(w) \leftarrow x$; CpFN.$\varrho_i(x) \leftarrow w$;
                        CpFN.$\beta_o(w) \leftarrow \kappa_o(u)$; CpFN.$\beta_i(w) \leftarrow$
                    $\kappa_i(u)$;
                    END-FOR
                    FOR (eachof $\forall\, \alpha \in \delta_o(u)$) DO
                        Call PROCEDURE H-
                    FRAGMENTATION(In $s \leftarrow \alpha$, CpFN);
                    END-FOR
                    exit();
                Case 'disjunctive-type (OR-split) activity':
                    $w \leftarrow u$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{w\}$;
                    CpFN.$\varrho_o(v) \leftarrow w$; CpFN.$\varrho_i(w) \leftarrow v$;
                    CpFN.$\beta_o(v) \leftarrow \kappa_o(s)$; CpFN.$\beta_i(v) \leftarrow \kappa_i(s)$;
                    FOR (eachof $\forall\, \alpha \in \delta_o(u)$) DO
                        Call PROCEDURE H-
                    FRAGMENTATION(In $s \leftarrow \alpha$, CpFN);
                    END-FOR
                    exit();
                Default: /* OR-join activity or AND-join activity */
                    $w \leftarrow u$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{w\}$;
                    CpFN.$\varrho_o(v) \leftarrow w$; CpFN.$\varrho_i(w) \leftarrow v$;
                    CpFN.$\beta_o(v) \leftarrow \kappa_o(s)$; CpFN.$\beta_i(v) \leftarrow \kappa_i(s)$;
                    break;
          END-SWITCH
          $s \leftarrow u$; $v \leftarrow w$;
        END-WHILE
        $w \leftarrow u$; CpFN.$\mathbf{A}^{cp} \leftarrow$ CpFN.$\mathbf{A}^{cp} \cup \{w\}$; /* $u$ is equal to $\alpha_F$. */
        CpFN.$\varrho_o(v) \leftarrow w$; CpFN.$\varrho_i(w) \leftarrow v$;
        CpFN.$\beta_o(v) \leftarrow \kappa_o(s)$; CpFN.$\beta_i(v) \leftarrow \kappa_i(s)$;
        PRINTOUT CpFNs
    END-PROCEDURE
```

2.3. Summaries

So far, as workflow fragmentation methods, the role-based workflow fragmentation method (Kim, 2012) (vertical fragmentation) and the controlpath-based workflow fragmentation method (Kim, 2012) (horizontal fragmentation) are introduced in this section. Additionally, we can easily imagine a hybrid fragmentation method by synthetically applying those two fragmentation methods. That is, it is possible to apply the role-based workflow fragmentation method to each of the controlpathbased workflow fragments, and then we can chop a structured workflow process model into a much larger number of fragments. Also, as an another vertical fragmentation method, it is naturally possible to break a workflow model by the name of actor-based fragmentation method, which is not described yet. The hybrid and the actor-based fragmentation methods ought to be much more suitable for those enterprise cloud workflow or the enterprise grid workflow enacting environments (Kim, 2007), in where a much larger number of workflow enactment components are needed to be involved.

3. Distributed workflow XML-event log format

After disseminating the workflow fragments (role-based fragments or controlpathbased fragments) that are vertically or horizontally (or in hybrid) fragmented by the corresponding fragmentation algorithm, the fragments are enacted by each of the distributed workflow engine' components. Then, as shortly explained in the previous section, the workflow engine's components that are taking a role of formatting events produce their event log messages after executing the requested services from the event triggering components the requester and the worklist handler. After doing the formatting job, they transfer the formatted event log messages to the event logging components the log agents, for example. Based on the formatted messages, the log agents form the XML-based event log information. The detailed names of the event types that are captured and logged by the mechanism are summarized as the followings, and they are represented as EventCode in the XML-based event log format (Kim, 2006).

- *Event Types*: Scheduled-Workitem, Started-Workitem, Completed-Workitem, Changed-Workitem-State

3.1. Workflow fragment event log

As a workflow fragment instance executes, a temporal execution sequence of its activities is produced and logged into a database or a file; this temporal execution sequence is called workflow fragment trace or temporal fragment, which is formally defined in [*Definition 5*]. The temporal fragment is made up of a set of fragment event logs as defined in the following [*Definition 4*].

Definition 4. Fragment Event Log. Let ***fel*** = (α, *pc*, *wf*, *f*, *ac*, *c*, ε, p^*, *t*, *s*) be a workflow fragment event, where α is a workitem (activity instance) number, ***pc*** is a package number, ***wf*** is a workflow process number, ***f*** is a fragment number, ***ac*** is an activity number, ***c*** is a workflow instance (case) number, ε is an event type, which is one of {Scheduled, Started, Completed}, ***p*** is a participant or performer, ***t*** is a timestamp, and ***s*** is a workitem state, which is one of {Inactive, Active, Suspended, Completed, Terminated, Aborted}. Note that * indicates multiplicity.

Table 3
Workflow fragment event log message and its XML-based language

Log Element	XML Tag	Description
FragmentLog	<FragmentLog>. . .</FragmentLog>	Event Log on Fragment
WorkitemID	<WorkitemID> WorkitemID </WorkitemID>	Workitem ID of the corresponding activity associated with the fragment
PackageID	<PackageID> PackageID </PackageID>	Package ID associated with the fragment
WorkflowID	<WorkflowID> WorkflowID </WorkflowID>	Workflow Process ID associated with the fragment
FragmentID	<FragmentID> FragmentID </FragmentID>	Workflow Fragment ID associated with the workitem
ActivityID	<ActivityID> ActivityID </ActivityID>	Activity ID associated with the fragment
WorkcaseID	<WorkcaseID> WorkcaseID </WorkcaseID>	Workcase ID associated with the fragment's instance
EventCode	<EventCode> EventCode = {AssignedWorkitem | StartedWorkitem | CompletedWorkitem | ChangedWorkitemState} </EventCode>	Event code performed by the workitem
EventTimestamp	<EventTimestamp> EvnetTimestamp </EventTimestamp>	The time happening the event code
Performer	<Performer> Performer </Performer>	Performer ID of the workitem
State	<State> State = {INACTIVE | ACTIVE | SUSPEND | COMPLETED | TERMINATED | ABORTED} </State>	The current state of the workitem

In general, we consider a workflow event log to be stored in an XML format. An XML-based workflow fragment event log language extended from (Kim, 2006) is precisely described in Table 3. Because of the page length limitation, now let's assume to simply use the language to describe the XML schema of a workflow fragment event log in this paper.

3.2. Workflow fragment traces (Temporal fragments)

Definition 5. Workflow Fragment Trace (Temporal Fragment). Let WFT(***f***) be the workflow fragment(***f***) instance's execution event trace, where WFT(***f***) = (fel_1 ,...,fel_n).

Especially, the workflow fragment trace is called temporal fragment, TF(***f***), if all activities of its underlined workflow fragment instance are successfully completed. There are three types of the temporal fragments according to the events type, Scheduled, Started, and Completed:

- ScheduledTime Temporal Fragment
  $\{fel_i \mid fel_i.\mathrm{c} = \mathbf{c} \wedge fel_i.f = \mathbf{f} \wedge fel_i.\mathrm{s} =$ '**Inactive**' $\wedge\ fel_i.\varepsilon=$'**ScheduledWorkitem**' $\wedge\ fel_i.t \leq fel_j.t \wedge i < j \wedge 1 \leq i, j \leq n\}$,
  which is a temporally ordered workflow fragment event sequence based upon the scheduled time-stamp.
- StartedTime Temporal Fragment
  $\{fel_i \mid fel_i.\mathrm{c} = \mathbf{c} \wedge fel_i.\mathrm{f} = \mathbf{f} \wedge fel_i.\mathrm{s} =$ '**Active**' $\wedge\ fel_i.\varepsilon =$ '**StartedWorkitem**' $\wedge\ fel_i.\mathrm{t} \leq fel_j.\mathrm{t} \wedge i < j \wedge 1 \leq i, j \leq n\}$,
  which is a temporally ordered workflow fragment event sequence based upon the started time-stamp.
- CompletedTime Temporal Fragment
  $\{fel_i \mid fel_i.\mathrm{c} = \mathbf{c} \wedge fel_i.\mathrm{f} = \mathbf{f} \wedge fel_i.\mathrm{s} =$ '**Completed**' $\wedge\ fel_i.\varepsilon =$ '**CompletedWorkitem**' $\wedge\ fel_i.t \leq fel_j.\mathrm{t} \wedge i < j \wedge 1 \leq i, j \leq n\}$,

which is a temporally ordered workflow fragment event sequence based upon the completed time-stamp.

As shown in the definition of temporal fragment, there are three types of temporal fragment ifferentiated from the temporal information (the event's timestamp) logged when the corresponding activity's workitem event was happened. Originally, in the workflow fragment event log schema, the events that are associated with the workitem are ScheduledWorkitem, StartedWorkitem, CompletedWorkitem, and ChangedWorkitemState. However, we take into account only three events—ScheduledWorkitem, StartedWorkitem, CompletedWorkitem—which have associated with their states, Inactive, Active and Completed, respectively, in the temporal fragment, in order to form the types of temporal fragment to be used in the distributed workflow process mining algorithm.

4. Distributed workflow process mining approaches

This section gives a distributed workflow process mining framework that eventually rediscovers a structured workflow process model from the distributed workflow fragments' execution event logs formatted in the temporal fragments of the previous section. The framework consists of a series of concepts and algorithms. However, we particularly focus on distributed workflow mining algorithms that are able to rediscover a structured workflow process model from a series of workflow fragment traces conceptually modeled by the concept of temporalworkcases. Finally, in order to prove the correctness of the algorithms, we show how it works for the simple structured workflow process model introduced in Fig. 2 and Table 1, as an example.

4.1. Framework

The distributed workflow process mining framework is illustrated in Fig. 5. The framework starts from the distributed workflow fragment event logs written in the XML-based workflow fragment event log language. The workflow event logging components residing in the distributed workflow enactment engines store all workflow fragments' execution event histories onto their own logging repositories. The workflow fragment event logs might be generated through the following three types of engine components:

- *Event triggering components* — Requester and Worklist Handler
- *Event formatting components* — Workcase and Object Pool
- *Event logging components* — Log Agent and Log File Storage

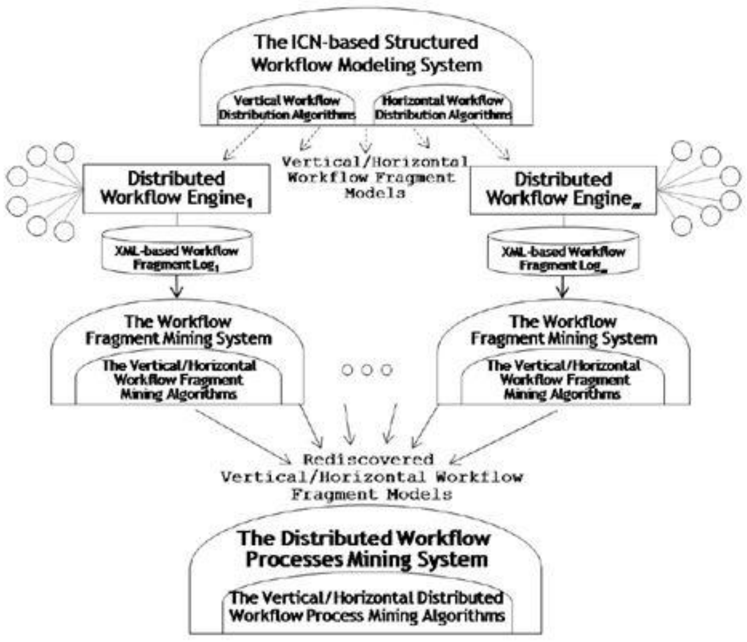

The ICN-based Structured Workflow Modeling System
Vertical Workflow Distribution Algorithms
Horizontal Workflow Distribution Algorithms
Vertical/Horizontal Workflow Fragment Models
Distributed Workflow Engine$_1$
Distributed Workflow Engine$_n$
XML-based Workflow Fragment Log$_1$
XML-based Workflow Fragment Log$_n$
The Workflow Fragment Mining System
The Vertical/Horizontal Workflow Fragment Mining Algorithms
Rediscovered Vertical/Horizontal Workflow Fragment Models
The Distributed Workflow Processes Mining System
The Vertical/Horizontal Distributed Workflow Process Mining Algorithms

Fig. 5. The distributed workflow process mining framework

The event triggering components handle the workflow fragment enactment services requested from the workflow clients, and the event formatting components try to compose the XML-based fragment event log messages after performing the requested services, and finally the event logging components, especially which is called the log agents, take in charge of the responsibility of the fragment event logging mechanism. Once each log agent receives the fragment event logs, and then transforms them into the XML-based fragment event log messages, and store the transformed messages onto the Log File Storage. From the XML-based workflow fragment event logs on each of the log file storages, it is possible to build a workflow fragment mining warehouse on each computing node of the distributed workflow systems, which has a cube with three dimensions, such as workflow fragment models, fragment instances, and activities. From the cube we extract a set of temporal fragments (traces) that is instantiated from a workflow fragment model. A temporal fragment is a temporal order of activity executions within an instance of the corresponding workflow fragment model, and it will be formally represented by a temporal fragment model. The details of the temporal fragment and its related models are precisely defined in the next section. Finally, the distributed workflow fragment rediscovery algorithm rediscovers a workflow fragment model by incrementally integrating a series of workflow fragment models, $\omega_1 \sim \omega_n$, one-by-one. The details of the algorithm and its operational example are described in the next sections, too.

Definition 6. Workflow Fragment Log and Warehouse. Let $I_i = \{c_1^i, ..., c_m^i\}$ be a set of completed workflow fragment instances (***m*** is the number of fragment instances) that have been instantiated from a vertical/horizontal workflow fragment model, WF_i. A workflow fragment warehouse consists of a set of workflow fragment logs, $WFL(I_1)$, ..., $WFL(I_n)$, where $WFL(I_i) = \forall WFT(c^i \in I_i)$, and ***n*** is the number of workflow fragment models managed in a single node of a distributed workflow system.

Based on these defined concepts, we are able to prepare the temporal fragments that become the input data of the workflow fragment mining algorithm proposed in this paper. Additionally, according to the types of temporal fragments, we can build three different types of workflow fragment logs and their warehouses as defined in [*Definition 6*]. Conclusively speaking, the workflow fragment mining algorithm may consider taking the temporal fragments, as in put data, coming from one of three workflow fragment warehouse types ScheduledTime-based Warehouse, StartedTime-based Warehouse, and CompletedTime-based Warehouse. Also, the algorithm may simultaneously take two types of temporal information such as ScheduledTime/CompletedTime or StartedTime/CompletedTime to rediscover workflow fragment models. In this case, the algorithm needs to take two types of the temporal fragments, each of which is belonged to its warehouse type, respectively. The algorithm presented in this paper will be taking care of the Started Time-based workflow fragment warehouse as the source of the temporal fragments. Nevertheless, it is sure for the algorithm to be able to be extended so as to handle two types of the temporal fragments as its input data.

4.2. Mining workflow fragment models

Actually, a workflow fragment mining system instantiated from the framework can be characterized by its underlying workflow fragment mining algorithm. Also, the system is depended on how to fragment the original structured workflow process model. If the original model is vertically fragmented, then the system works with the vertical workflow fragment mining algorithm, and the system works with the horizontal workflow fragment mining algorithm if it is horizontally fragmented. Also, the workflow fragment logs and warehouses can be organized by the type of the fragmentation, too. This paper tries to deploy a possible horizontal fragmentation-based approach for rediscovering the horizontal workflow fragment models, and a possible vertical fragmentation-based approach for mining the vertical workflow fragment models, as well.

4.2.1. Horizontal temporal fragments

The detailed procedure for rediscovering the horizontal workflow fragment models from the distributed workflow fragment logs is illustrated in Fig. 6. As shown in Fig. 4, the horizontal workflow fragments of the sample structured workflow process model introduced in the previous section are modeled and distributed. So, based on their enactment event logs, two kinds of the horizontal fragment logs are created under the control of their own distributed workflow enactment components. From each of the horizontal fragment logs, it is possible to make groups of horizontal temporal fragments, as shown in the middle part of Fig. 6. Each of the horizontal temporal fragment groups can be formally represented by a horizontal workflow fragment model defined in [*Definition 7*], and it also can be graphically modeled as shown in the right-most part of Fig. 6. The primary reason we use the formal representation of the horizontal workflow fragment model is just because it is surely convenient in composing the structured workflow process mining algorithm.

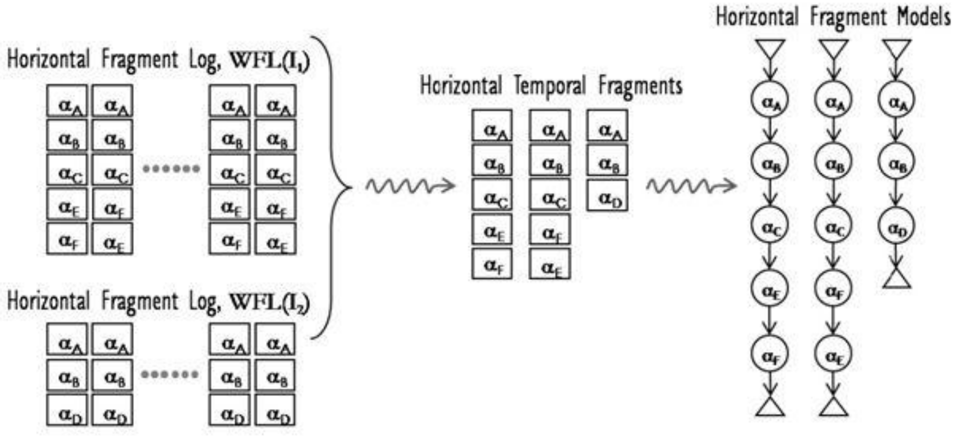

Horizontal Fragment Log, WFL(I1)
Horizontal Fragment Log, WFL(I2)
Horizontal Temporal Fragments
Horizontal Fragment Models

Fig. 6. Horizontal workflow fragment models

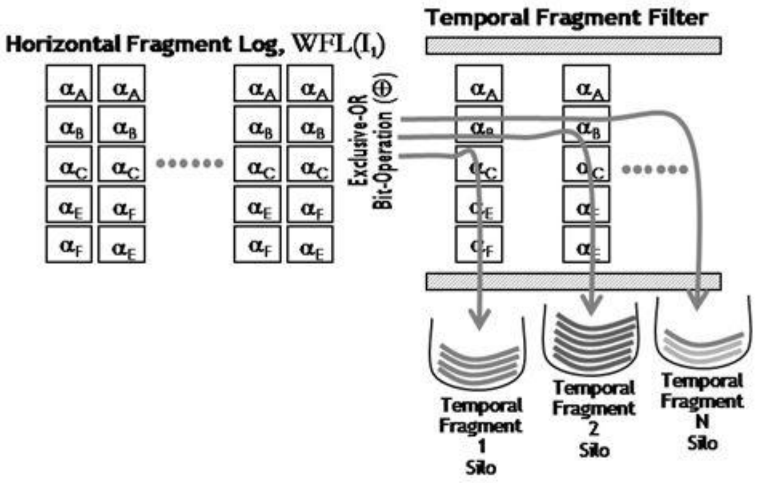


Fig. 7. Filtering horizontal workflow fragment logs

Definition 7. Horizontal Workflow Fragment Model (HWFM). A horizontal workflow fragment model is formally defined through 3-tuple **HWF** = (ω, **P**, **S**) over an activity set **A**, where

- **P** is a predecessor activity of some external workcase model, which is connected into the current fragment model;
- **S** is a successor activity of some external workcase model, which is connected from the current fragment model;
- $\omega = \omega_i \cup \omega_o$,

where, $\omega_o : \mathbf{A} \rightarrow \wp(\alpha \in \mathbf{A})$ is a single-valued mapping function of an activity to its immediate successor in a horizontal temporal fragment, and $\omega_i : \mathbf{A} \rightarrow \wp(\alpha \in \mathbf{A})$ is a single-valued mapping function of an activity to its immediate predecessor in a horizontal temporal fragment.

One possible solution for mining the horizontal temporal fragment groups from the horizontal fragment logs is to use a filtering approach with multiple layers, as illustrated in Fig. 7. Each layer in the multiple layered filter is to be incrementally built whenever a mismatched temporal fragment is passed through the filter's layers. The figure shows a conceptual idea for the horizontal workflow fragment mining algorithm. The details of the algorithm won't be described in this paper because of the page limitation.

4.2.2. Vertical temporal fragments

The detailed procedure for mining the vertical workflow fragment models from the distributed workflow fragment logs is illustrated in Fig. 8. As shown in Fig. 3, the vertical workflow fragments of the sample structured workflow process model introduced in the previous section are modeled and distributed. So, based on their enactment event logs, three kinds of the vertical fragment logs are created under the control of their own distributed workflow enactment components. From each of the vertical fragment logs, it is possible to make groups of vertical temporal fragments, as shown in the 2nd step of Fig. 8 that illustrates a stepwise approach for rediscovering vertical fragment models. Each of the vertical temporal fragment groups can be formally represented by a vertical workflow fragment model defined in [*Definition 8*], and it also can be graphically modeled as shown in the 3rd part of Fig. 8. As you can see in the stepwise approach of Fig. 8, the group of vertical fragment models with a same instance ID will be eventually matched with one of the horizontal fragment models, which is exactly same to one of the possible control-paths of the original structured workflow model. As a necessary consequence, it must be definitely possible to rediscover the exactly same horizontal temporal fragment models from the vertical fragment logs.

Conclusively speaking, the vertical fragmentation-based mining approach is able to rediscover all possible controlpaths (Park & Kim, 2008), each of which is exactly same to each of the horizontal temporal fragment models, and finally which are used to rediscover the original structured workflow process model by the workflow process mining algorithm to be presented in the next section.

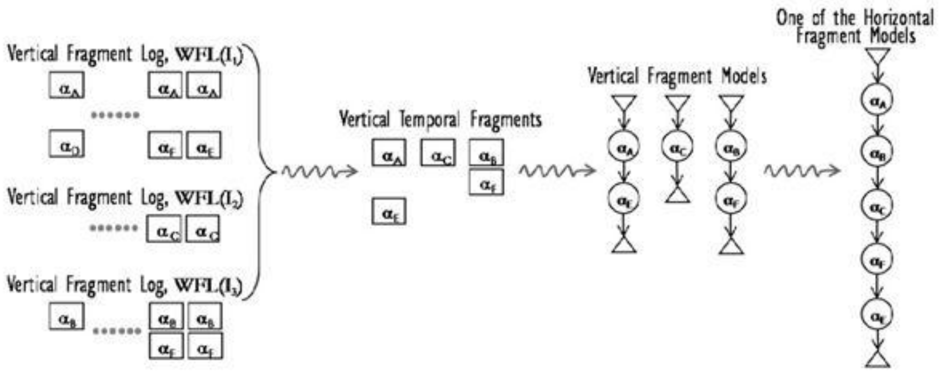

Vertical Fragment Log, WFL(I1)
Vertical Fragment Log, WFL(I2)
Vertical Fragment Log, WFL(I3)
Vertical Temporal Fragments
Vertical Fragment Models
One of the Horizontal Fragment Models

Fig. 8. Vertical workflow fragment models

Definition 8. Vertical Workflow Fragment Model (VWFM). A vertical workflow fragment model is formally defined through 3-tuple **VWF** = (υ, **P**, **S**) over an activity set **A**, where

- **P** is a predecessor activity of some external workcase model, which is connected into the current fragment model;
- **S** is a successor activity of some external workcase model, which is connected from the current fragment model;
- $\upsilon = \upsilon_i \cup \upsilon_o$,

where, υ_o : $\mathbf{A} \rightarrow \wp(\alpha \in \mathbf{A})$ is a single-valued mapping function of an activity to its immediate successor in a vertical temporal fragment, and υ_i : $\mathbf{A} \rightarrow \wp(\alpha \in \mathbf{A})$ is a single-valued mapping function of an activity to its immediate predecessor in a vertical temporal fragment.

Also, one possible solution for mining the vertical temporal fragment groups from the vertical fragment logs is to use the filtering approach illustrated in Fig. 7. Each layer in the multiple layered filter is to be incrementally built whenever a mismatched vertical temporal fragment is passed through the filter's layers.

4.2.3. Mining structured workflow process models

This section gives a full detail of the distributed workflow process mining algorithm and demonstrates the algorithm's correctness though an example with the sample structured workflow process model. More specifically speaking, the algorithm will build up one reasonable structured workflow process model by amalgamating those horizontal/vertical temporal fragment groups that are rediscovered from the distributed workflow fragment logs by the previous horizontal/vertical workflow fragment mining algorithms. Each of the horizontal/vertical temporal fragment groups is embodied in a horizontal/vertical workflow fragment model as explained in the previous section. In summary, the central ideas of the algorithm are the followings:

- The algorithm repeatedly modifies a temporarily rediscovered structured workflow process model by incorporating a new horizontal/vertical workflow fragment (HWF/VWF) model until running out all models. Thus, it is an incremental algorithm: after seeing the first HWF/VWF model the algorithm generates a reasonable structured workflow process model for that HWF/VWF model and upon seeing the second HWF/VWF model, it amalgamates the new HWF/VWF model into the existing reasonable model.
- The algorithm is a series of rewrite operations that transform the reasonable model plus HWF/VWF model into a new reasonable model until bringing up to the last. As a consequence of these amalgamating operations, the final reasonable model becomes the structured workflow process model rediscovered from the horizontally or vertically distributed workflow fragment logs.

4.3.1. The basic rediscovery principles

As described in the previous section, a structured workflow process model is designed through the three types of control transitions sequential, disjunctive and conjunctive transition with keeping the matched pair and proper nesting properties. Therefore, the horizontal/vertical distributed workflow mining algorithm must be obligated to rediscover these transitions by amalgamating the horizontal temporal fragment models rediscovered from the horizontal/vertical fragment logs. The basic idea of the amalgamation procedure conducted by the algorithm is to incrementally amalgamate one horizontal/vertical fragment model after another. Also, during the amalgamation procedure works, the most important thing is to observe and seek those three types of transitions.

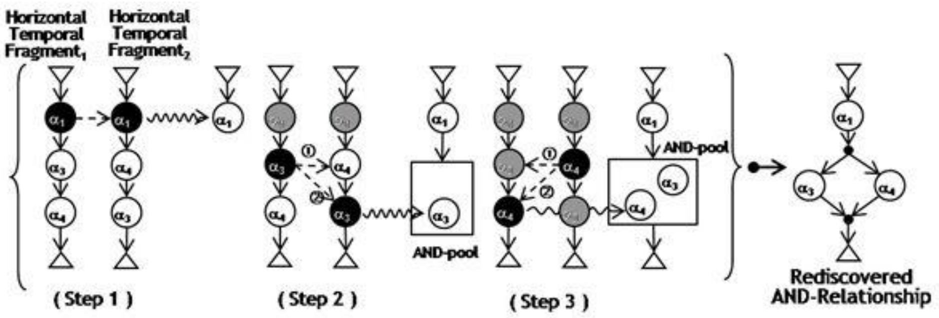

Horizontal Temporal Fragment$_1$
Horizontal Temporal Fragment$_2$
AND-pool
AND-pool
(Step 1)
(Step 2)
(Step 3)
Rediscovered AND-Relationship

Fig. 9. The rediscovery principle of AND-transition

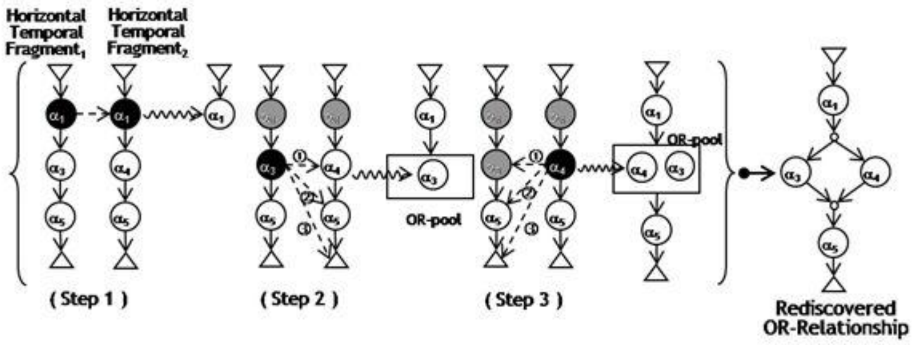

Horizontal Temporal Fragment$_1$
Horizontal Temporal Fragment$_2$
OR-pool
OR-pool
(Step 1)
(Step 2)
(Step 3)
Rediscovered OR-Relationship

Fig. 10. The rediscovery principle of OR-transition

Precisely, basic amalgamating principles seeking each of the transition types are as follows: if a certain activity is positioned at the same temporal order in all horizontal/vertical fragment models, then the activity is to be involved in a sequential transition; else if the activity is at the different temporal order in some horizontal/vertical fragment models, then we can infer that the activity is to be involved in a conjunctive transition; otherwise if the activity is either presented in some horizontal/vertical fragment models or not presented in the other horizontal/vertical fragment models, then it has got to be involved in a disjunctive transition. As simple examples of the amalgamating principles, Fig. 9 and Fig. 10 algorithmically illustrate the amalgamation procedures rediscovering a conjunctive transition and a disjunctive transition, respectively.

4.3.2. *Distributed workflow process mining algorithm*

Based upon the basic rediscovery principles, we conceive a distributed workflow process mining algorithm in order to rediscover a reasonable structured workflow process model by amalgamating the horizontal/vertical workflow fragment models. Because of the page limitation this section would not make a full description of the algorithm. However, we just introduce the detailed algorithm as follows, which is pseudo-coded as detail as possible with some explanations in comments, so that one is able to easily grasp the algorithm without the full description.

PROCEDURE DistributedMining():

```
1:  Input : A Set of horizontal/vertical Workflow Fragment Models, ∀(hwf / vwf [i], i = 1..m);
2:          where, hwf / vwf [1] == START(▽), hwf / vwf [m] == END(△);
3:  Output : (1) A Rediscovered Structured Workflow Process Model (SWPM), R = ($\delta$, $\kappa$, I, O);
4:          - The Activity Set of SWPM, A = {$\alpha_1 .. \alpha_n$}, (hwf[i], i = 1..m) ∈ A;
5:        (2) A Set of Horizontal/vertical Workflow Fragment Models (HWFMs/VWFMs), ∀HWF = ($\omega$, P, S) or VWF = ($\upsilon$, P, S);
6:
7:  Initialize : $\delta_i$(START(▽)) ← {NULL};
8:          $\delta_o$(END(△)) ← {NULL};
9:  PROCEDURE DistributedMining()
10: BEGIN
11:   WHILE ((hwf [] ← readOneWorkcase()) ≠ EOF) DO
12:     i ← 1;
13:     WHILE (hwf [i] ≠ END(△)) DO
14:         $\omega o$ / $\upsilon o$ (hwf / vwf [i]) ← hwf / vwf [i + 1]; i ← i + 1; $\omega i$ (hwf [i]) ← hwf [i − 1]; or $\upsilon i$ (vwf [i]) ← vwf [i − 1];
15:     END WHILE
16:     * Rediscovering the temporary RWPM from the current WCM */
17:     FOR (i = 1; i < m; i++) DO
18:         IF (Is $\delta o$ (hwf [i]) an empty-set?) THEN
19:             $\delta o$ (hwf [i]) ← $\omega o$ (hwf [i]); or $\delta o$ (vwf [i]) ← $\upsilon o$ (vwf [i]);
20:             continue;
21:         END IF
22:         IF (isANDTransition(hwf / vwf [i], $\omega o$ / $\upsilon o$ (hwf / vwf [i])) == TRUE) THEN
23:             continue;
24:         END IF
25:         FOR (each set, $\alpha$, of sets in $\delta o$ (hwf / vwf [i])) DO
26:             SWITCH (checkupTransition($\alpha$, $\omega o$ / $\upsilon o$ (hwf / vwf [i])) DO
27:                 Case 'fixed transition':
28:                 Case 'sequential relationship':
29:                     $\delta o$ (hwf / vwf [i]) ← $\omega o$ / $\upsilon o$ (hwf / vwf [i]);
30:                     break;
31:                 Case 'conjunctive transition (AND-split)':
32:                     ANDset ← makeANDTransition($\alpha$, $\omega o$ / $\upsilon o$ (hwf / vwf [i]));
33:                     $\delta o$ (hwf / vwf [i]) ← $\delta o$ (hwf / vwf [i]) ∪ ANDset;
34:                     eliminatePreviousTransition($\alpha$, $\omega o$ / $\upsilon o$ (hwf
                              / vwf [i]));
35:                              **break;**
36:                          **Case 'disjunctive transition (OR-split)':**
37:                              ORset ← **makeORTransition**($\alpha$, $\omega_O$ / $\upsilon_O$ (hwf / vwf [i]));
38:                              $\delta_O$ (hwf / vwf [i]) ← $\delta_O$ (hwf / vwf [i]) $\cup$ ORset;
39:                              **eliminatePreviousTransition**($\alpha$, $\omega_O$ / $\upsilon_O$ (hwf / vwf [i]));
40:                              **break;**
41:                          **Default:** /* Exceptions */
42:                              :**printErrorMessage();**
43:                              **break;**
44:                      **END SWITCH**
45:              **END FOR**
46:          **END FOR**
47:      **END WHILE**
48:      **finishupTheRediscoveredModel();** /* with its input-activity sets, ($\delta_i$ (hwf /vwf [i]), i = 1..n) and its transition-conditions */
49:      $\delta_i$ ($\alpha_1$) ← {START($\bigtriangledown$)}; $\delta_O$ ($\alpha_n$) ← {END($\vartriangle$)};
50:      **PRINTOUT**
51:          (1) The Rediscovered Structured Workflow Process Model, SWPM, **R** = ($\delta$, $\kappa$, **I**, **O**);
52:          (2) A Set of the Horizontal/Vertical Workflow Fragment Models, HWFs/VWFs, ∀**HWF** = ($\omega$, **P, S**) or **VWF** = ($\upsilon$, **P, S**);
53: **END PROCEDURE**
```

4.3.3. Constraints of the algorithm

As emphasized in the previous sections, this algorithm is operable on the concept of horizontally/vertically distributed structured workflow process model that retains the proper nesting and matched pair properties (Kim & Ellis, 2007). Keeping these properties causes to constrain the algorithm as well as the modeling work; nevertheless, it might be worthy to preserve the constraints because they can play a very important role in increasing the integrity of the workflow model. Additionally, not only the improperly nested workflow model makes its analysis complicated, but also the workflow model with unmatched pairs may be stuck and run into a deadlock situation during its runtime execution. Another important issue in designing horizontally/vertically distributed work-flow process mining algorithms is about how to handle loop transitions in a structured workflow process model, because they may produce not only a lot of workflow event logs but also much more complicated patterns of temporal fragments. Precisely, according to the number of repetitions and the inside structure of a loop transition, the model's execution may generate very diverse and complicated patterns of temporal fragments. Therefore, the algorithm pro-posed in this paper has got to be extended in order to properly handle the loop transitions. We would leave this issue to our future research work.

5. Related works

So far, there have been several workflow mining related researches and developments in the workflow literature. Some of them have proposed the algorithms van der Aalst et al., 2003; Schimm, 2004; Pinter & Golani, 2004; Kim & Ellis, 2006; Agrawal, Gunopulos, & Leymann, 1998; de Medeiros, van Dongen, van der Aalst, & Weijters, 2004; Ellis, Kim, & Rembert, 2006; Silva, Zhang, & Shanahan, 2005; Gaaloul & Godart, 2005; Kim & Ellis, 2007; Kim, 2009) for workflow mining functionality, and others have developed the workflow mining systems and tools (Herbst & Karagiannis, 2004; Kim, 2005). Particularly, as the first industrial application of the workflow mining, J. Herbsta and D. Karagiannisb in (Herbst & Karagiannis, 2004) presented the most important results of their experimental evaluation and experiences of the InWoLvE workflow mining system. However, almost all of the contribution are still focusing on the development of the basic functionality of workflow mining techniques. Especially, W.M.P. van der Aalst's research group, through the papers of (van der Aalst et al., 2003; de Medeiros et al., 2004; Medeiros & Weijters, 2005), proposed the fundamental definition and the use of workflow mining to support the design of workflows, and described the most challenging problems and some of the workflow mining approaches and algorithms. Also, Clarence Ellis's research group newly defined the scope of workflow mining concept from the view point of that workflow systems are "people systems" that must be designed, deployed, and understood within their social and organizational contexts. Thus, they argue in (Kim & Ellis, 2006; Ellis, Kim, & Rembert, 2006; Ellis et al., 2006; Kim & Ellis, 2007; Kim, 2009) that there is a need to expand the concept of workflow discovery beyond the process dimension to encompass multidimensional perspective such as social, organizational, and informational perspectives; as well as other perspectives. This paper is the partial result of the collaborative research on mining the workflow's multidimensional perspectives, and also it would be the pioneering result in the distributed workflow process mining issues.

6. Conclusion

This paper proposed the distributed structured workflow process mining approaches rediscovering a structured workflow process from the distributed workflow fragment logs. The approach is based on the structured workflow process model designed by the information control net workflow modeling methodology, and it conceived the workflow fragmentation techniques such as vertical fragmentation, horizontal fragmentation and hybrid. Finally, This paper showed that it is able to properly handle the distributed workflow fragments logs with the enactment event histories of the three different types of control transitions—sequential, conjunctive and disjunctive transitions—on the horizontally/vertically fragmented workflow processes through the horizontally/vertically distributed workflow fragment logs, as example. Also, the proposed approaches need to be extended to cope with the loop-structured distributed workflow models in the near future. In a consequence, according for the distributed enterprise computing environments like enterprise grid/P2P and enterprise cloud computing environments to be hot-issued in the literature, distributed workflow process mining methodologies and systems are rapidly growing and coping with a wide diversity of domains in terms of their applications and working environments. So, the literature needs various, advanced, and specialized distributed workflow mining techniques and architectures. We strongly believe that this work might be one of those impeccable attempts and pioneering contributions for improving and advancing the distributed workflow fragmentation and mining technology.

Acknowledgements

This research was supported by the National Research Foundation's Basic Research Grant, No. 2012-006971, South Korea. Also, this paper is extended from the paper (Kim, 2009) published in IWDXP2009.

References

Agrawal, R., Gunopulos, D., & Leymann, F. (1998). Mining process models from workflow logs. *Proceeding of Sixth International Conference on Extending Database Technology* (pp. 469–483).

de Medeiros, A. K. A., van Dongen, B. F., van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). *Process mining: Extending the α-algorithm to mine short loops.* BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven.

Ellis, C. A. (1979). Information control nets: A mathematical model of office information flow. *Proceedings of the ACM Conference on Simulation, Measurement and Modeling of Computer Systems* (pp. 225–240). Boulder, Colorado.

Ellis, C. A., Kim, K.-H., & Rembert, A. J. (2006). Workflow mining: Definitions, techniques, and future directions. In L. Fischer (ed.) *Workflow Handbook* (pp. 213–228). Lighthouse Point: Future Strategies.

Ellis, C. A., Rembert, A. J., Kim, K.-H., & Wainer, J. (2006). Beyond worflow mining. *Proceedings of the 5th International Business Process Management Conference* (BPM'06).

Gaaloul, W., & Godart, C. (2005). Mining workflow recovery from event based logs. *Lecture Notes in Computer Science, 3649*, 169–185.

Herbst, J. & Karagiannis, D. (2004). Workflow mining with InWoLvE. *Computers in Industry, 53*(3), 245–264.

Kim, K. (2005). A workflow trace classification mining tool. *International Journal of Computer Science and Network Security, 5*(11), 19–25.

Kim, K. (2006). A XML-based workflow event logging mechanism for workflow mining. *Lecture Notes in Computer Science, 3842*, 132–136.

Kim, K., & Ellis, C. (2006). Workflow reduction for reachable-path rediscovery in workflow mining. *Series of Studies in Computational Intelligence: Foundations and Novel Approaches in Data Mining, 9*, 289–310.

Kim, K.-H. (2007). A layered workflow knowledge Grid/P2P architecture and its models for future generation workflow systems. *Future GenrationComputer Systems, 23*(3), 304–316.

Kim, K., & Ellis, C. A. (2007). σ-Algorithm: Structured workflow process mining through amalgamating temporal workcases. *Lecture Notes in Artificial Intelligence, 4426*, 119–130.

Kim, K. (2009). Mining workflow processes from XML-based distributed workflow event logs. *IEEE Proceedings of International Workshop on Distributed XML Processing: Theory and Practice* (pp. 587–594), Austria.

Kim, K. (2012). A model-driven workflow fragmentation framework for collaborative workflow architectures and systems. *Journal of Network and Computer Applications, 35*(1), 97–110.

Liu, R., & Kumar, A. (2005). An analysis and taxonomy of unstructured workflows. *Lecture Notes in Computer Science, 3649*, 268–284.

Medeiros, A. K. A. D., & Weijters, A. J. M. M. (2005). Genetic process mining. *Lecture Notes in Computer Science, 3536*, 48–69.

Park, M., & Kim, K. (2008). Control-path oriented workflow intelligence analyses. *Journal of Information Science and Engineering, 24*(2), 343–359.

Park, M. J., & Kim, K. (2010). A workflow event logging mechanism and its implications on quality of workflows. *Journal of Information Science and Engineering, 26*, 1817–1830.

Pinter, S. S., & Golani, M. (2004). Discovering workflow models from activities' lifespans. *Computers in Industry, 53*(3), 283–296.

Schimm, G. (2004). Mining exact models of concurrent workflows. *Computers in Industry, 53*(3), 265–281.

Silva, R., Zhang, J., & Shanahan, J. G. (2005). Probabilistic workflow mining. *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 275–284).

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. J. M. M. (2003).Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering, 47*(2), 237–267.