International Journal of Innovative Computing, Information and Control
Volume **8**, Number **6**, June **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **4263–4284**

AN ICN-BASED WORKFLOW PROCESS REDISCOVERY FRAMEWORK

Kwanghoon Pio Kim

Collaboration Technology Research Lab.
Department of Computer Science
Kyonggi University
Suwonsi, Kyonggido 443-760, South Korea
kwang@kgu.ac.kr

Received January 2011; revised May 2011

Abstract. *Workflow management systems help execute, monitor, and manage work process flow and execution. These systems, as they are executing, keep a record of who does what and when (i.e., log of events). The activity of using computer software to examine these records and derive various structural data results is termed workflow rediscovery. It has encompassed behavioral (process/control-flow), social, informational (data-flow), and organizational perspectives of a workflow model. Particularly, this paper*[1] *focuses on rediscovering the behavioral aspect of an ICN-based workflow model*[2] *from the workflow enactment histories (log of events). We term this activity workflow process rediscovery. That is, this paper proposes a formalized framework from the logging activity to the algorithmic activity of the workflow process rediscovery, termed an ICN-based workflow process rediscovery framework complete with a series of formal definitions and their related algorithms. The essential algorithm, σ-algorithm, is able to handle the basic behavioral primitives (sequential, alternative, parallel) and the loop behavioral primitive through incrementally amalgamating a series of temporal workcases that are temporal orders of the associated workflow activities' enactment events. This paper precisely describes analytic estimation and performance analysis of the algorithm.*

Keywords: Workflow management system, Events log, Workflow process mining and rediscovery, Temporal workcase, Information control net, Workflow process rediscovery framework

1. **Introduction.** A workflow management system (WfMS) is defined as a system that (fully or partially) automates the definition, creation, execution, and management of work processes through the use of software that is able to interpret the defined processes, interact with workflow participants, and invoke the use of IT tools and applications. Steps of a work process are termed activities, and jobs flowing through the system are termed workcases or workflow instances. Such a WfMS and its related technologies have been increasingly deployed and are hot-issues in the enterprise information systems' arena. This has seen a boom in workflows modeling and reengineering works, and became a catalyst to trigger the emergence of the workflow mining/rediscovering concept. This concept rediscovers several perspectives—control flow, data flow, social, and organizational perspectives—of workflow models from workflow execution and event histories collected at runtime. Furthermore, as the real-world's workflow models have been closely associated with e-Commerce, ERP (Enterprise Resource Planning), KM (Knowledge Management) and CRM (Customer Relationship Management), the workflow models have become larger

[1]This paper is an extended version of the paper published in [18] by adding more than 30 percent substantial new contributions.

[2]A workflow definition model that is represented by the basic concept of Information Control Net [10].

and much more complex in their behavioral structures. Therefore, the workflow rediscovery functionality has led to much more pragmatic requirements in the literature.

In general, a workflow model is described by the specific entities, such as activity, role, actor, invoked applications and relevant data, and particularly, it specifies its behavioral procedure through the transition precedences—sequential, parallel (AND), alternative (OR) and iterative (LOOP) execution sequences—among the associated activities. The paper uses the information control net (ICN) [10] to represent workflow models in terms of the workflow modeling methodology. Especially it confines workflow process models to emphasize the control (or behavioral) perspective of the workflow model. In addition, suppose that the workflow process model keeps the proper-nesting and the matched-pairing properties in modeling the parallel, alternative, and iterative transitions—AND-split/AND-join, OR-split/OR-join and LOOP-split/LOOP-join nodes, and it is eventually composed into an ICN-based structured workflow process model [18]. Particularly, this paper focuses on rediscovering the behavioral (process, control flow) patterns from the event log histories of workflow instances (workcases) spawned by ICN-based structured workflow models. That is, this paper proposes an ICN-based workflow process rediscovery framework that provides a systematic means for rediscovering structured workflow process models from logs of workflow enactment events happened through the activities' execution, based upon the concept of the ICN-based workflow process model. Note that a workflow enactment event log is typically an interleaved list of events from numerous workcases—workflow instances. This is represented by XML-based format and language defined in this paper. The proposed framework is furnished with several well-defined concepts, principles, and algorithms including the σ-*Algorithm* developed by the author in [18]. We can detect the temporal order of activity executions for each workcase, which is termed a temporal workcase by examining the log; and then infer the general structure of the underlying workflow process model, in terms of the mining principle. Particularly, we assume that the general structures of the models imply those models from the typical sequential, parallel and alternative transitions.

As a simple example, suppose we examine the execution log of a workflow process model of four activities, a_1, a_2, a_3 and a_4. Suppose also that all four activities are always executed in some order by every workcase. If we observe over a large number of workcases that a_1 is always executed first and a_4 is always executed last, then we can begin to piece together a workflow process model that requires a_1 to complete before all other activities, and a_4 to execute after all others. If we find some workcases in the log where a_2 begins before a_3, and other cases where a_2 begins after a_3, then we can infer that the workflow process begins with a_1; after it completes, a_2 and a_3 execute concurrently (parallel transition: AND control flow); and after they both complete, a_4 executes. This is an extremely simplified example that ignores the other important control transition constructs—alternative and iterative transitions (OR control flow and LOOP control flow)—and their combinations. Conclusively, the framework proposed in this paper formalizes the overall rediscovery procedure coping with not only the essential rediscovery algorithm but also the XML-based event log formats and their transformation algorithms.

The main section of this paper will show that the framework and its related algorithms are able to handle all of the possible activity execution cases via the concepts of temporal workcases. Finally, the paper analytically discusses the workflow process rediscovery framework and surveys its related works.

2. **Structured Workflow Process Model.**

This paper uses the information control net methodology [10] to represent wokflow 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, precedence, applications, and repositories. It has been used within real offices as well as hypothetical automated offices to yield a comprehensive description of activities, to test the underlying office description for certain flaws and inconsistencies, to quantify certain aspects of office information flow, and to suggest possible office restructuring permutations. Particularly this section focuses on the activities and their related information flows by newly defining the basic concept of structured workflow process model [16], preserving proper-nesting and matched-pairing properties, through its graphical and formal representations.

2.1. **Graphical representation.** As shown in Figure 1, a workflow process model consists of a set of activities connected by control orderings represented by activity transitions. This is, it is a predefined set of work steps, called activities, and a partial ordering (or control flow) of these activities. Activities can be related to each other 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). An activity is classified into either a compound activity containing another subprocess, or a basic unit of work termed an elementary activity. The elementary activity denotes being executed in one of three modes: manual, automatic, or hybrid.

Figure 2 shows a simple structured workflow process model. The model consists of six activities, $\alpha_1 \sim \alpha_6$, and the activities are linked to each other through combinations of the basic transition types. Note that the AND-Control nodes (AND-split and AND-join), and the OR-Control nodes (OR-split and OR-join) in the model are not only properly nested but also pair-matched to build a structured workflow process model.

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

Definition 2.1. Structured Workflow Process Model (SWPM). *A basic structured workflow process model is formally defined through 4-tuple* $\Gamma = (\delta, \kappa, \boldsymbol{I}, \boldsymbol{O})$ *over an activity set* $\boldsymbol{A}$*, and a transition-condition set* $\boldsymbol{T}$*, where*

- $\boldsymbol{I}$ *is a finite set of initial input repositories, assumed to be loaded with information by some external process before execution of the model;*
- $\boldsymbol{O}$ *is a finite set of final output repositories, which is containing information used by some external process after execution of the model;*

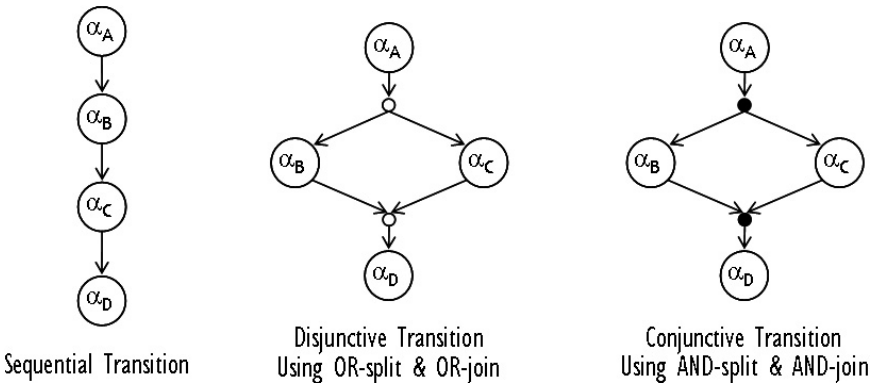

FIGURE 1. Graphical notations for the basic transition types

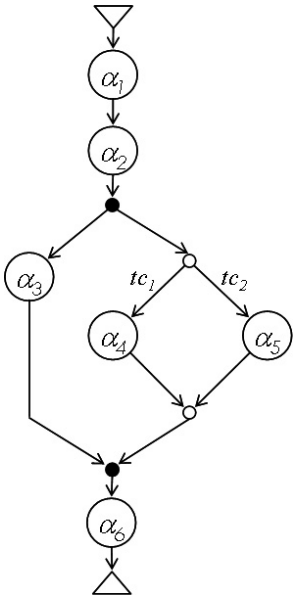

FIGURE 2. A simple structured workflow process model

- $\delta = \delta_i \cup \delta_o$,
  where $\delta_o : \boldsymbol{A} \longrightarrow \wp(\boldsymbol{A})$ *is a multi-valued mapping function of an activity to its set of (immediate) successors, and* $\delta_i : \boldsymbol{A} \longrightarrow \wp(\boldsymbol{A})$ *is a multi-valued mapping function of an activity to its set of (immediate) predecessors;*
- $\kappa = \kappa_i \cup \kappa_o$,
  where $\kappa_i : \boldsymbol{A} \longrightarrow \wp(\boldsymbol{A})$ *is a multi-valued mapping function of an activity to its incoming transition-conditions* $(\subseteq \boldsymbol{T})$ *on each arc,* $(\delta_i(\alpha), \alpha)$*; and* $\kappa_o : \boldsymbol{A} \longrightarrow \wp(\boldsymbol{A})$*: is a multi-valued mapping function of an activity to its outgoing transition-conditions* $(\subseteq \boldsymbol{T})$ *on each arc,* $(\alpha, \delta_o(\alpha))$.

Starting and Terminating Nodes. Additionally, the execution of a structured workflow process model commences with a single λ transition-condition. So, we always assume without loss of generality that there is a single starting node (α_I). At the commencement, it is assumed that all input repositories in the set $\boldsymbol{I}$ have been initialized with data by the external system:

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

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

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

Structured Modeling Methodology. The role of the structured modeling methodology is to preserve the proper-nesting and the matched-pairing properties in modeling a workflow process model. Conclusively, the formal definition implies that the temporal ordering of a structured workflow process model can be interpreted as follows: for any activity α $(\delta = \delta_i \cup \delta_o)$, in general,

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

$$\{\beta_{n1}, \beta_{n2}, \ldots, \beta_{nm(n)}\}$$
$$\}$$

denotes 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 in which 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. In addition, if $m(i) = 1$ 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. Also,

$$\delta_i(\alpha) = \{$$
$$\{\beta_{11}, \beta_{12}, \ldots, \beta_{1m(1)}\},$$
$$\{\beta_{21}, \beta_{22}, \ldots, \beta_{2m(2)}\},$$
$$\ldots,$$
$$\{\beta_{n1}, \beta_{n2}, \ldots, \beta_{nm(n)}\}$$
$$\}$$

denotes 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 dots—AND-join, and the latter, that an activity has a disjunctive (or decision) incoming transition, is represented by a hollow dots—OR-join.

In summary, a structured workflow process model is constructed by the structured modeling methodology to preserve the proper-nesting and the matched-pairing properties. Its formal definition implies that the temporal orderings of the model can be interpreted as the ordered combination of the following formal transition types, each of which corresponds to each of the basic transition types graphically depicted in Figure 1. Table 1 represents the formal description of the structured workflow process model of Figure 2.

: Sequential Transition
$incoming \rightarrow \delta_i(\alpha_B) = \{\{\alpha_A\}\}$; $outgoing \rightarrow \delta_o(\alpha_B) = \{\{\alpha_C\}\}$;
: OR Transition
$or\text{-}split \rightarrow \delta_o(\alpha_A) = \{\{\alpha_B\}, \{\alpha_C\}\}$; $or\text{-}join \rightarrow \delta_i(\alpha_D) = \{\{\alpha_B\}, \{\alpha_C\}\}$;
: AND Transition
$and\text{-}split \rightarrow \delta_o(\alpha_A) = \{\{\alpha_B, \alpha_C\}\}$; $and\text{-}join \rightarrow \delta_i(\alpha_D) = \{\{\alpha_B, \alpha_C\}\}$;

3. **Framework Rediscovering Structured ICN-Workflow Processes.** In this section, we propose a workflow process rediscovery framework that mines a structured workflow process model from the temporal workcases filtered from the workflow execution events log. The framework is made up of a series of concepts and algorithms. However, we particularly focus on the mining algorithm, termed σ-*Algorithm*, and its directly related concept—temporal workcase. Finally, we show how it works for a typical structured workflow process model, as an example, comprising the three basic transition types to prove the correctness of the algorithm.

3.1. **Framework.** Figure 3 illustrates the workflow process rediscovery framework. The framework starts from the event logs written in XWELL (XML-based Workflow Event

TABLE 1. Formal representation of the structured workflow process model

$\Gamma = (\delta, \kappa, I, O)$ over A, T /* The Structured Workflow Process Model		
$A = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_I, \alpha_F\}$ /* Activities		
$T = \{d(default), tc_1, tc_2\}$ /* Transition Conditions		
$I = \{\emptyset\}$ /* Initial Input Repositories		
$O = \{\emptyset\}$ /* Final Output Repositories		
$\delta = \delta_i \cup \delta_o$	$\delta_i(\alpha_I) = \emptyset;$	$\delta_o(\alpha_I) = \{\{\alpha_1\}\};$
	$\delta_i(\alpha_1) = \{\{\alpha_I\}\};$	$\delta_o(\alpha_1) = \{\{\alpha_2\}\};$
	$\delta_i(\alpha_2) = \{\{\alpha_1\}\};$	$\delta_o(\alpha_2) = \{\{\alpha_3, \{\{\alpha_4\}, \{\alpha_5\}\}\}\};$
	$\delta_i(\alpha_3) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_3) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_4) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_4) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_5) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_5) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_6) = \{\{\alpha_3, \{\{\alpha_4\}, \{\alpha_5\}\}\}\};$	$\delta_o(\alpha_6) = \{\{\alpha_F\}\};$
	$\delta_i(\alpha_F) = \{\{\alpha_6\}\};$	$\delta_o(\alpha_F) = \emptyset;$
$\kappa = \kappa_i \cup \kappa_o$	$\kappa_i(\alpha_I) = \emptyset;$	$\kappa_o(\alpha_I) = \{d\};$
	$\kappa_i(\alpha_1) = \{d\};$	$\kappa_o(\alpha_1) = \{d\};$
	$\kappa_i(\alpha_2) = \{d\};$	$\kappa_o(\alpha_2) = \{d, tc_1, tc_2\};$
	$\kappa_i(\alpha_3) = \{d\};$	$\kappa_o(\alpha_3) = \{d\};$
	$\kappa_i(\alpha_4) = \{tc_1\};$	$\kappa_o(\alpha_4) = \{d\};$
	$\kappa_i(\alpha_5) = \{tc_2\};$	$\kappa_o(\alpha_5) = \{d\};$
	$\kappa_i(\alpha_6) = \{d\};$	$\kappa_o(\alpha_6) = \{d\};$
	$\kappa_i(\alpha_F) = \{d\};$	$\kappa_o(\alpha_F) = \emptyset;$

Log Language) [7], by which the workflow event logging mechanism of a workflow enactment engine stores all workflow process execution event histories triggered by the engine components. XWELL needs to be standardized such that heterogeneous workflow mining systems are able to collect the event logs without any additional data transformations. In general, the event logs might be produced by the three types of workflow engine's components, such as event triggering components, event formatting components, event logging components.

The *event triggering components* handle the workflow enactment services requested from workflow clients. These services can be categorized into three levels of classification — Workcase level class, Running activity level class, and Workitem level class. The *event formatting components* try to compose event log messages according to the service classes after performing the requested services. Finally, the *event logging components*, especially the log agents, are responsible for the event logging mechanism. Once a log agent receives event logs and transforms them into XML-based log messages. It stores the transformed messages into the Log File Storage.

Based on the XML-based event logs on the log file storage, we can build a workflow process mining warehouse that forms a three dimensional cube with dimensions, workflow process models, temporal workcases, and activities. We extract a set of temporal workcases (traces) that is instantiated from a structured workflow process model from the cube. A temporal workcase is a temporal order of activity executions within an instance of the corresponding workflow process model. It will formally represent a workcase model. The details of the temporal workcase and its related models are precisely defined in the next section. Finally, the structured workflow process mining algorithm rediscovers a structured workflow process model by incrementally amalgamating a series of workcase

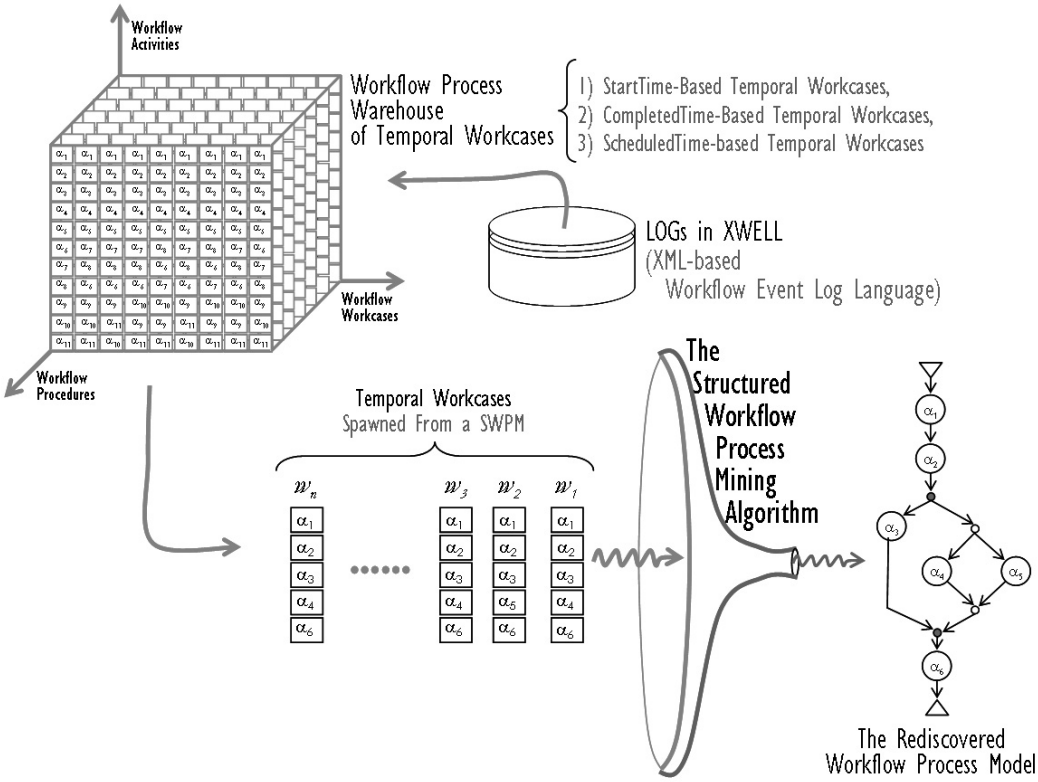

FIGURE 3. The workflow process rediscovery framework

models, $\omega_1 \sim \omega_n$, one-by-one. The details of the algorithm and its operational example are described in the next sections.

3.2. **σ-algorithm.** This section gives full detail to the structured workflow process mining algorithm and demonstrates the algorithm's correctness using an example structured workflow process model. We have to assume that there might be many and possibly infinite workflow process models (if fake activities are allowed) that could be mined from a set of traces to mine a structured workflow process model if we use one of the conventional algorithms like [2,3,19]. Even though some of these models are very easy to compute, others are not. Thus, we must pick one reasonable model out of the infinite number of models as a final output of the algorithm. However, those traditional algorithms have high-orders of complexity. Therefore, we must take a fundamentally different approach to conceive an algorithm. More specifically, our algorithm will build up one *reasonable* model by amalgamating one trace after another, each of which is embodied in a workcase model. In summary, the central idea of our approach is as follows:

- *The algorithm repeatedly modifies a temporarily rediscovered workflow process model, termed reasonable model, by incorporating one trace at a time into it until it runs out of traces.*
- *Thus, it is an incremental algorithm; after seeing the first trace the algorithm generates a new reasonable model, and upon seeing the second trace it merges into the existing reasonable model, and so forth.*
- *Conclusively, the algorithm is made up of a series of rewrite operations that transform the reasonable model plus one trace into a new reasonable model until the last. The final reasonable model becomes the structured workflow process model rediscovered from all traces of the corresponding workflow process model.*

3.2.1. *Workflow traces: temporal workcases.* A temporal execution sequence of its activities is produced and logged into a database or a file, as a workflow process instance executes; this temporal execution sequence is termed *workflow trace* or *temporal workcase*, formally defined in Definition 3.2. The temporal workcase is made up of a set of *workflow event logs* as defined in Definition 3.1. In addition, we would define the concept of *workflow process log* in Definition 3.3, which is produced from a set of temporal workcases spawned from a single structured workflow process model.

Definition 3.1. Workflow Event Log. *Let* $\boldsymbol{we} = (\alpha, pc, wf, c, ac, \varepsilon, p^*, t, s)$ *be a workflow event, where* α *is a workitem (activity instance) number,* $\boldsymbol{pc}$ *is a package number,* $\boldsymbol{wf}$ *is a workflow process number,* $\boldsymbol{c}$ *is a workflow instance (workcase) number,* $\boldsymbol{ac}$ *is an activity number,* ε *is an event type, which is one of* {*Scheduled, Started, Changed, Completed*}, $\boldsymbol{p}$ *is a participant or performer,* $\boldsymbol{t}$ *is a timestamp, and* $\boldsymbol{s}$ *is an activity state, which is one of* {*Inactive, Active, Suspended, Completed, Terminated, Aborted*}. *Note that * indicates multiplicity.*

In general, we consider a workflow event log to be stored in XML format. An XML-based workflow event log language has been studied and proposed in [7] for workflow mining. There are three levels of classes—workitem class, running activity class, and workcase class—for the workflow event logs. Here, we just introduce the XML schema of a workflow event log for the workitem class, as shown in Table 2, to properly illustrate the basic concept of the workflow mining. The WORKITEM attribute of a workflow event represents a workitem ID that is uniquely assigned by the workflow enactment engine when it is corresponding activity is scheduled to the worklist handler. In addition, the corresponding activity is identified by the buildtime's IDs—PACKAGE ID, WORKFLOW ID and ACTIVITY ID—and the runtime's ID—WORKCASE ID—that represents a process instance maintained and executed by a runtime client. The EVENT attribute is used to specify the event type—Scheduled, Started, Changed, Completed—of the activity instance being logged. The PARTICIPANT attribute is used to specify who performs the workitem. The TIMESTAMP attribute specifies the time the event occurred. Finally, the STATE attribute represents the workitem's runtime state maintained by the engine. Whenever the workitem's state is changed, it is logged with the eventcode, WMChangedWorkitmeState. The possible state is one of those states, such as INACTIVE, ACTIVE, SUSPENDED, COMPLETED, TERMINATED and ABORTED.

Definition 3.2. Workflow Trace (Temporal Workcase). *Let* $WT(\boldsymbol{c})$ *be the workflow trace of process instance* $\boldsymbol{c}$*, where* $WT(\boldsymbol{c}) = (we_1, \ldots, we_n)$. *Especially, the workflow trace is termed a temporal workcase,* $TW(\boldsymbol{c})$*, if all activities of its underlined process instance are successfully completed. The three types of temporal workcases according to the event types are Scheduled, Started, Completed:*

- *ScheduledTime Temporal Workcase*
  $\{we_i | we_i.c = \boldsymbol{c} \wedge we_i.e = \textit{'Scheduled'} \wedge we_i.t \leq we_j.t \wedge i < j \wedge 1 \leq i, j \leq n\}$, *which is a temporally ordered workflow event sequence based upon the scheduled time stamp.*
- *StartedTime Temporal Workcase*
  $\{we_i | we_i.c = \boldsymbol{c} \wedge we_i.e = \textit{'Started'} \wedge we_i.t \leq we_j.t \wedge i < j \wedge 1 \leq i, j \leq n\}$, *which is a temporally ordered workflow event sequence based upon the stated time stamp.*
- *CompletedTime Temporal Workcase*
  $\{we_i | we_i.c = \boldsymbol{c} \wedge we_i.e = \textit{'Completed'} \wedge we_i.t \leq we_j.t \wedge i < j \wedge 1 \leq i, j \leq n\}$, *which is a temporally ordered workflow event sequence based upon the completed time stamp.*

The three types of temporal workcases are differentiated from the temporal information (the event's timestamp) logged when the corresponding activity's workitem event

TABLE 2. XML schema of the workitem class's event logs

Log Element	XML Tag		Description
WorkitemLog	<WorkitemLog> ... </WorkitemLog>		Event Log on Workitem
WorkitemID	<WorkitemID> </WorkitemID>	`WorkitemID`	Unique ID for the workitem
PackageID	<PackageID> </PackageID>	`PackageID`	Package ID identifying the package being associated with this workitem
WorkflowID	<WorkflowID> </WorkflowID>	`WorkflowID`	Process Definition ID identifying the workflow being associated with this workitem
WorkcaseID	<InstanceID> </WorkflowID>	`WorkcaseID`	Workcase (Process Instance) ID identifying the workcase being associated with this workitem
ActivityID	<ActivityID> </WorkflowID>	`ActivityID`	Activity Definition ID for the base activity of this workitem
EventCode	<EventCode> `= { WMAssginedWorkitem | WMTakenWorkitem | WMChangedWorkitemState | WMCompletedWorkitem }` </EventCode>	`EventCode`	This message code is associated with the events of this workitem
EventTimestamp	<EventTimestamp> </EventTimestamp>	`EventTimestamp`	Timestamp at the time when the event was recorded
Performer	<Performer>	`Performer` </Performer>	Performer ID of the current workitem
State	<State>	`State = { INACTIVE | ACTIVE | SUSPENDED | COMPLETED | TERMINATED | ABORTED }` </State>	Current state of the workitem

occurred, as shown in the definition of temporal workcase. The events that are associated with the workitem are `WMAssginedWorkitem`, `WMTakenWorkitem`, `WMChangedWorkitemState` and `WMCompletedWorkitem`, based upon the workflow event log format of Table 2. However, we take into account only three events—`WMAssginedWorkitem`, `WMTakenWorkitem` and `WMCompletedWorkitem`, which correspond to `Scheduled`, `Started` and `Completed`, respectively, in the temporal workcase, to form the types of temporal workcases to be used in the workflow mining algorithm.

Definition 3.3. Workflow Process Log and Warehouse. *Let* $I_i = \{c_1^i, \ldots, c_m^i\}$ *be a set of completed process instances (*$\boldsymbol{m}$ *is the number of process instances) that have been instantiated from a workflow process model,* I_i*. A workflow process warehouse consists of a set of workflow process logs,* $WL(I_1)$*, ...,* $WL(I_n)$*, where* $WL(I_i) = \forall WT(c^i \in I_i)$*, and* $\boldsymbol{n}$ *is the number of workflow process models managed in a system.*

We are able to prepare the temporal workcases, which become the input data of the workflow mining algorithm proposed in this paper, based on these defined concepts. Additionally, we can build three different types of workflow process logs and their warehouses as defined in Definition 3.3, according to the types of temporal workcases. The workflow mining algorithm may consider taking the temporal workcases, as input data, coming

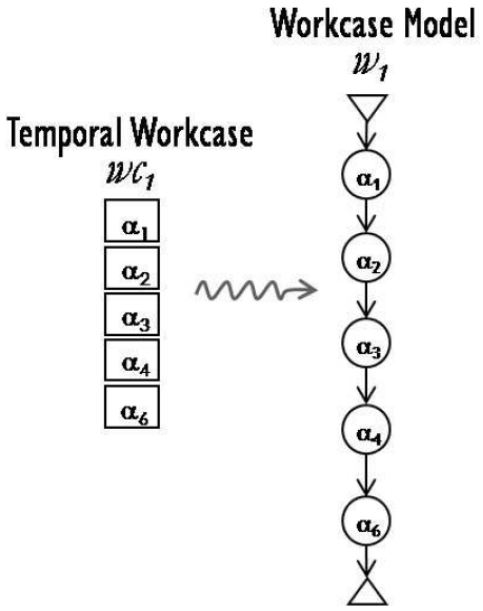

FIGURE 4. The workcase model

from one of three workflow process warehouse types—ScheduledTime-based Warehouse, StartedTime-based Warehouse, and CompletedTime-based Warehouse. In addition, the algorithm may simultaneously take two types of temporal information such as ScheduledTime/CompletedTime or StartedTime/CompletedTime to rediscover structured workflow process models. In this case, the algorithm needs to take two types of the temporal workcases, each of which belongs to its respective warehouse type. The algorithm presented in this paper will take care of the StartedTime-based workflow process warehouse as the source of the temporal workcases. Nevertheless, the algorithm will be able to be extended to handle two types of temporal workcases as its input data.

3.2.2. *Workcase model.* Each of the temporal workcases, as the input data of the algorithm, is represented to a workcase model via a series of converting operations of the algorithm. We formally define the workcase model in Definition 3.4. It can be graphically represented, as shown in Figure 4. The primary reason we use the formal workcase model is due to its convenience in composing the structured workflow process mining algorithm.

Definition 3.4. Workcase Model (WCM). *A workcase model is formally defined through 3-tuple* $\boldsymbol{W} = (\omega, \boldsymbol{P}, \boldsymbol{S})$ *over an enacted-activity set* $\boldsymbol{E}(\subseteq \boldsymbol{A})$ *out of the original ICN-based workflow model, where*

- $\boldsymbol{P}$ *is a predecessor activity of some external workcase model that is connected to the current workcase model;*
- $\boldsymbol{S}$ *is a successor activity of some external workcase model that is connected from the current workcase model;*
- $\omega = \omega_i \cup \omega_o$,
  where, $\omega_o : \boldsymbol{E} \longrightarrow \wp(\boldsymbol{E})$ *is a single-valued mapping function of an activity to its immediate successor in a temporal workcase, and* $\omega_i : \boldsymbol{E} \longrightarrow \wp(\boldsymbol{E})$ *is a single-valued mapping function of an activity to its immediate predecessor in a temporal workcase.*

3.2.3. *The basic amalgamating principles.* As described in the previous section, a structured workflow process model is designed by the three types of control transitions—sequen-

tial, disjunctive and conjunctive transition—keeping the matched-pairing and proper-nesting properties. Therefore, the structured workflow process mining algorithm must be obligated to rediscover these transitions by amalgamating the temporal workcases of a workflow process log. The basic idea of the amalgamation procedure conducted by the algorithm is to incrementally amalgamate one workcase model after another. In addition, the most important thing is to observe and seek those three types of transitions during the amalgamation procedure task.

Precisely, the basic amalgamating principles seeking each of the transition types are as follows: if a certain activity is positioned in the same temporal order in all workcase models, then the activity is to be involved in a sequential transition; else if the activity is in a different temporal order in some workcase 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 workcase models or not presented in the other workcase models, then it has to be involved in a disjunctive transition.

We algorithmically illustrate the amalgamation procedures rediscovering a conjunctive transition and a disjunctive transition through Figure 5 and Figure 6, respectively, as simple examples of the amalgamating principles. In Figure 5, suppose we examine the workflow process log of a structured workflow process model that has three activities, a_1, a_3 and a_4, and try to amalgamate two specific workcase models; the temporal order of a_3 and a_4 in one workcase model, is reversed on the other workcase model. Therefore, we can infer that the activities, a_3 and a_4, are involved in a conjunctive transition of the structured workflow process.

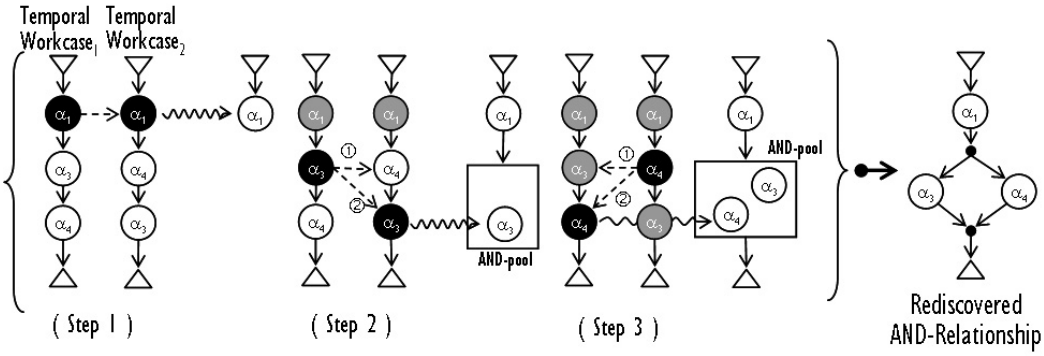

FIGURE 5. The amalgamating principle rediscovering AND-transition

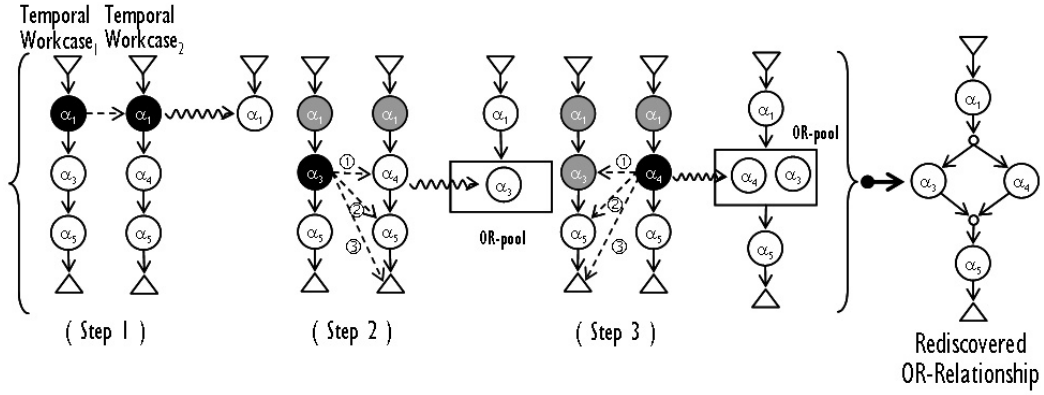

FIGURE 6. The amalgamating principle rediscovering OR-transition

In Figure 6, we also assume that we examine the workflow process log of a structured workflow process model that has four activities, a_1, a_3, a_4 and a_5, and try to amalgamate two specific workcase models; the temporal order of a_1 and a_5 in one workcase model is the same in the other workcase model; in addition, the positions of a_3 and a_4 in the temporal order are the same in these two workcase models respectively. Conversely, the activities, a_3 and a_4, are not presented in these two workcase models simultaneously. Therefore, we can infer that the activities, a_3 and a_4, are involved in a disjunctive transition of the structured workflow process.

3.2.4. *The SWP mining algorithm.* We conceive a workflow mining algorithm to rediscover a reasonable structured workflow process model from a workflow process log, based upon the basic amalgamating principles. We name it *σ-Algorithm* [18], because its basic idea is to incrementally amalgamate the temporal workcase, which simply reflects the conceptual idea of the summation operator ($\sum$) in mathematics. We could not fully describe the algorithm here. However, we introduce the simple descriptions of a series of operational functions used in the algorithm as follows. In addition, we give the detailed algorithm with the following pseudo-codes with explanations as comments, so it is easy to grasp the algorithm without a full description.

- **readOneWorkcase()** composes a temporal workcase model, $wc[\ ]$, from a workflow trace with its event history logs.
- **isANDTransition()** makes a decision whether two activities (input parameters of the function) on the current temporal workcase model have an AND-relationship, or not.
- **checkupTransition()** decides the relationship out of "fixed, sequential, conjunctive, and disjunctive relationships" between an activity set (the first input parameter) on the current rediscovered SWPM and an activity (the seconde input parameter) on the new temporal workcase model.
- **makeANDTransition()** amalgamates the input parameter's activity into the current temporary rediscovered SWPM with a conjunctive (AND) relationship.
- **makeORTransition()** amalgamates the input parameter's activity into the current temporary rediscovered SWPM with a disjunctive (OR) relationship.
- **eliminatePreviousTransition()** eliminates the input parameter's activity from the current temporal workcase model after composing AND or OR relationships on the current temporary rediscovered SWPM.

PROCEDURE SWPMiningProcedure():

1: **Input :** A Set of Temporal Workcases, $\forall(wc[i], i = 1..m)$;
2: $where, wc[1] ==$ START($\bigtriangledown$), $wc[m] ==$ END($\bigtriangleup$);
3: **Output :** (1) A Rediscovered **SWPM**, $\mathbf{R} = (\delta, \kappa, \mathbf{I}, \mathbf{O})$;
4: - The Activity Set of SWPM, $\mathbf{A} = \{\alpha_1\ ..\ \alpha_n\}, (wc[i], i = 1..m) \in \mathbf{A}$;
5: (2) A Set of Workcase Models (**WCMs**), $\forall\ \mathbf{W} = (\omega, \mathbf{P}, \mathbf{S})$;
6:
7: **Initialize :** δ_i(START($\bigtriangledown$)) $\leftarrow$ {NULL}; δ_o(END($\bigtriangleup$)) $\leftarrow$ {NULL};
8:
9: **PROCEDURE SWPMiningProcedure()**
10: **BEGIN**
11: **WHILE** $((wc[\ ] \leftarrow$ **readOneWorkcase**$()) \neq$ **EOF**) **DO**
12: $i \leftarrow 1$;
13: **WHILE** $(wc[i] \neq$ END($\bigtriangleup$)) **DO**
14: $\omega_o(wc[i]) \leftarrow wc[i+1]$; $i \leftarrow i+1$; $\omega_i(wc[i]) \leftarrow wc[i-1]$;
15: **END WHILE**

```
16:     /* Rediscovering a temporary Rediscovered SWPM from the current
        WCM */
17:     FOR (i = 1; i < m; i++;) DO
18:         IF (Is δ_o(wc[i]) an empty-set?) THEN
19:             δ_o(wc[i]) ← ω_o(wc[i]); continue;
20:         END IF
21:         IF (isANDTransition(wc[i], ω_o(wc[i])) == TRUE) THEN
22:             continue;
23:         END IF
24:         FOR (each set, a, of sets in δ_o(wc[i])) DO
25:             SWITCH (checkupTransition(a, ω_o(wc[i])) DO
26:               Case 'fixed transition':
27:               Case 'sequential relationship':
28:                   δ_o(wc[i]) ← ω_o(wc[i]);
29:                   break;
30:               Case 'conjunctive transition (AND-split)':
31:                   ANDset ← makeANDTransition(a, ω_o(wc[i]));
32:                   δ_o(wc[i]) ← δ_o(wc[i]) ∪ ANDset;
33:                   eliminatePreviousTransition(a, ω_o(wc[i]));
34:                   break;
35:               Case 'disjunctive transition (OR-split)':
36:                   ORset ← makeORTransition(a, ω_o(wc[i]));
37:                   δ_o(wc[i]) ← δ_o(wc[i]) ∪ ORset;
38:                   eliminatePreviousTransition(a, ω_o(wc[i]));
39:                   break;
40:               Default: /* Exceptions */
41:                   printErrorMessage();
42:                   break;
43:             END SWITCH
44:         END FOR
45:     END FOR
46:   END WHILE
47:   finishupSWPM();
48:   /*Its input-activity sets, (δ_i(wc[i]), i = 1..n) and its transition-conditions*/
49:   δ_i(α_1) ← {START(▽)}; δ_o(α_n) ← {END(△)};
50:   PRINTOUT
51:     (1) The Rediscovered Structured Workflow Process Model, SWPM, R = (δ, κ,
        I, O);
52:     (2) A Set of the Workcase Models, WCMs, ∀W = (ω, P, S);
53: END PROCEDURE
```

3.2.5. *An operational example.* Finally, Figure 7 algorithmically illustrates the algorithm's operational example. The right-hand side of the figure, which is formally defined in Table 3, is the rediscovered structured workflow process model that the algorithm extracts from the temporal workcases; these are the typical four temporal workcases that can be possibly produced from the original structured workflow process model. As you may expect, the algorithm is not concerned with the original model; nevertheless, we need it to generate a set of temporal workcases and to verify the algorithm. Fortunately, we are able to imagine that the original model produces the following four *StartedTime temporal workcases*:

(1) $a_1 \rightarrow a_2 \rightarrow a_4 \rightarrow a_3 \rightarrow a_6$

(2) $a_1 \rightarrow a_2 \rightarrow a_5 \rightarrow a_3 \rightarrow a_6$
(3) $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_6$
(4) $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_5 \rightarrow a_6$.

Figure 7 illustrates the detailed amalgamating procedure and its associated data structures that show that the algorithm incrementally amalgamates the *StartedTime temporal workcases* one-by-one. In the figure, a group of linked lists under each of the temporal workcases represents the temporal orders, $\omega_o(\alpha \in \mathbf{E})$, of the corresponding temporal workcase model. We assume that start and end activities are omitted from each of the temporal workcases. For example, the first group of the linked lists in the figure comes from the first temporal model with start and end activities, (1) start$\rightarrow a_1 \rightarrow a_2 \rightarrow a_4 \rightarrow a_3 \rightarrow a_6 \rightarrow$end. And, the second group of the linked lists is the result of amalgamating the first temporal workcase model with the second temporal workcase model, (2) start$\rightarrow a_1 \rightarrow a_2 \rightarrow a_5 \rightarrow a_3 \rightarrow a_6 \rightarrow$ end; where, $\otimes$ represents the amalgamating operator. Similarly, amalgamating all of those four temporal workcases produces the fourth group of the linked lists, which eventually builds up into the rediscovered structured workflow process model in the right-hand side of the figure.

3.2.6. *Analysis of the algorithm.* As emphasized in the previous sections, this algorithm operates on the concept of a structured workflow process model that retains the proper-nesting and matched-pairing properties. Keeping these properties constrain the modeling workflow processes and the mining workflow processes; nevertheless, it might be worthy to

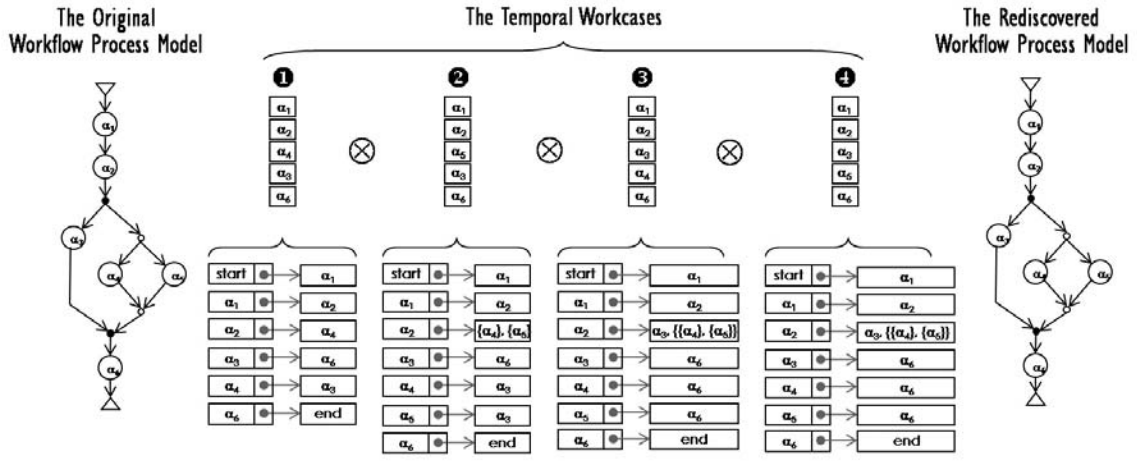

FIGURE 7. An operational example of the algorithm

TABLE 3. Control flows of the rediscovered structured workflow process model

$\mathbf{R} = (\delta, \kappa, I, O)$ /* The Rediscovered Structured Workflow Process Model		
$\delta = \delta_i \cup \delta_o$	$\delta_i(\alpha_I) = \emptyset;$	$\delta_o(\alpha_I) = \{\{\alpha_1\}\};$
	$\delta_i(\alpha_1) = \{\{\alpha_I\}\};$	$\delta_o(\alpha_1) = \{\{\alpha_2\}\};$
	$\delta_i(\alpha_2) = \{\{\alpha_1\}\};$	$\delta_o(\alpha_2) = \{\{\alpha_3, \{\{\alpha_4\}, \{\alpha_5\}\}\}\};$
	$\delta_i(\alpha_3) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_3) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_4) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_4) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_5) = \{\{\alpha_2\}\};$	$\delta_o(\alpha_5) = \{\{\alpha_6\}\};$
	$\delta_i(\alpha_6) = \{\{\alpha_3, \{\{\alpha_4\}, \{\alpha_5\}\}\}\};$	$\delta_o(\alpha_6) = \{\{\alpha_F\}\};$
	$\delta_i(\alpha_F) = \{\{\alpha_6\}\};$	$\delta_o(\alpha_F) = \emptyset;$

preserve the constraints because they can play an important role in increasing the integrity of workflow models. Especially, an improperly nested workflow model complicates its analysis and the workflow model with unmatched pairs may be stuck and reach a deadlock during its execution. In view of the results so far analyzed, we would say that the proper-nesting and matched-pairing properties might no longer become constraints in the mining algorithms and the workflow modeling works.

Another important issue in designing workflow process mining algorithms is how to handle loop transitions in a workflow process model, because they may produce many workflow event logs and much more complicated patterns of temporal workcases. The model's execution may generate very diverse and complex patterns of temporal workcases based on the number of repetitions and the inside structure of a loop transition. Therefore, the algorithm proposed in this paper must be extended to correctly handle the loop transitions. We will leave this issue to our future research.

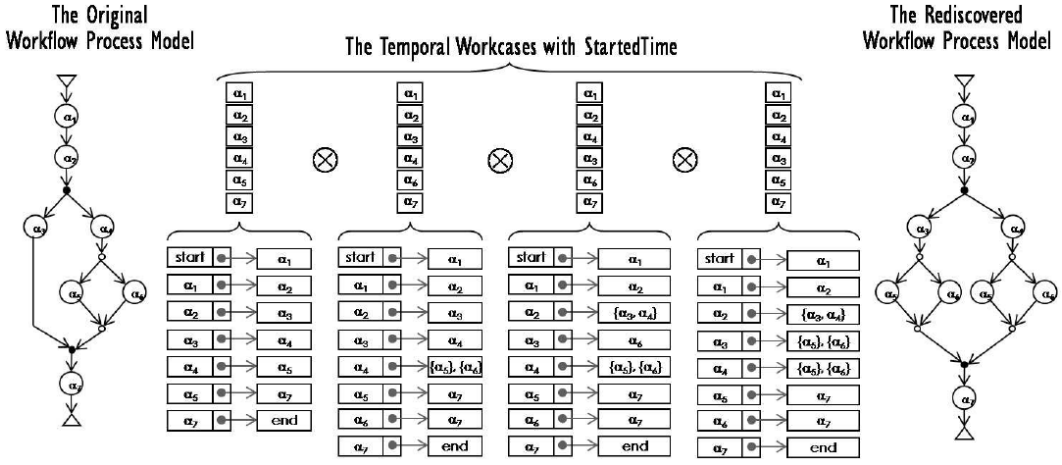

FIGURE 8. The algorithm's abnormality with *StartedTime Temporal Workcases*

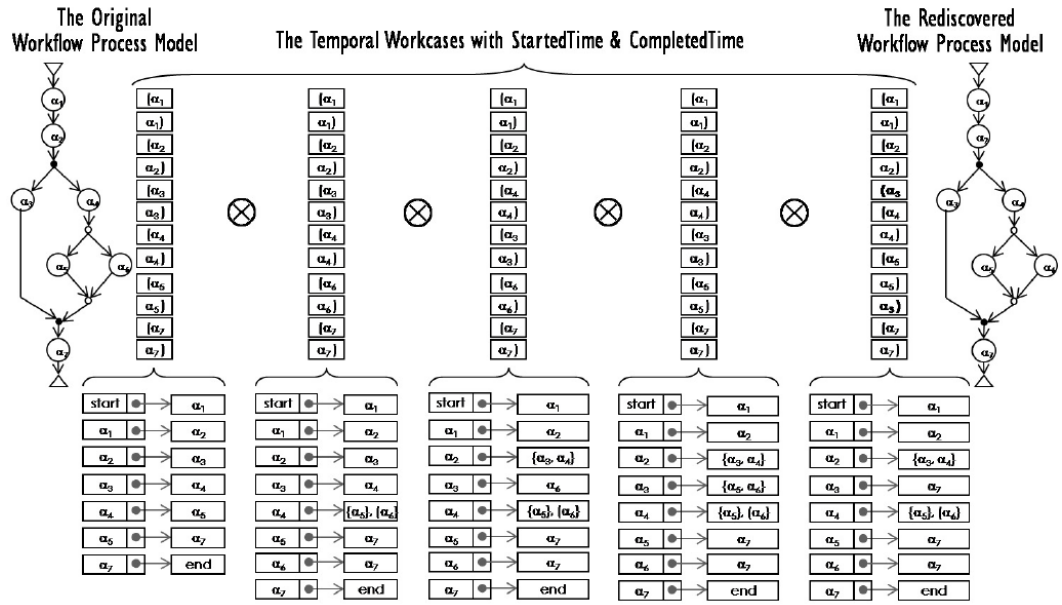

FIGURE 9. The resolution of the abnormality with *StartedTime and CompletedTime Temporal Workcases*

Additionally, the σ-*Algorithm* presented in the paper may not guarantee rediscovering the exact structured workflow process model, in terms of the algorithms' correctness issue, because the algorithm is based on the only single type of temporal information—*StartedTime* temporal workcases. As illustrated in Figure 8, the algorithm may rediscover a kind of strange workflow process models duplicating activities on OR constructs or AND constructs, which might not happen in the real modeling situation, and so which might be an abnormal structured workflow process model. This duplicated activities problem comes from using only one type of temporal information in the σ-*Algorithm*, like in the case of the example temporal workcases Figure 9. It is true that the rediscovered model is semantically correct; however, it is syntactically incorrect, as you see. Therefore, the σ-*Algorithm* is able to resolve the abnormality by using more sophisticated temporal information. That is, Figure 9 shows that the duplicate activities problem has been resolved by supplementing the *StartedTime* temporal orders and the *CompletedTime* temporal orders in logging the activities' execution events.

Finally, in terms of the complexity issue, the σ-*Algorithm*'s time complexity is $\bigcirc(\mathsf{N} \times \mathsf{M})$, where N is the number of activities associated in the original workflow process model, and M is the number of temporal workcases. Fortunately, it is possible to dramatically reduce the number of temporal workcases, because all of the temporal workcases filtered from the workflow process log (warehouse) can be grouped by the workflow reachable-pathes [5] of the original workflow process model.

4. **Implementation of the Framework.**

The author's research group has recently completed the development of a workflow management system that aims for very large scale workflows applications, and it is dubbed e-Chautauqua workflow management system [25]. e-Chautauqua is based on the workcase-oriented workflow architecture [26], and especially we have implemented it by the Enterprize Java Beans framework approach, while almost all conventional workflow systems are based on the activity-oriented workflow architecture proposed by OMG [27-30]. This section shortly introduces e-Chautauqua's event logging mechanism and formats. As a functional part of the proposed framework, we have developed an XML-based event logging mechanism and language, which is abbreviated to XWELL. In this section, we describe the functional structure of log agents, and explain about how the engine components take events, generate the events' log message formats and their language, and finally store them on log database. Additionally, we introduce the asynchronous logging message queue mechanism that is used for the engine components to store their event log information formatted in XWELL.

4.1. **Engine's logging components.**

The core engine components of e-Chautauqua's engine are the workcase objects residing on the workcase pool, as shown in Figure 10. That is, the control flow management is done by the workcase objects in the workcase-oriented workflow engine (e-Chautauqua), and so the activity precedence information is stored to the inside of each workcase object as data. Figure 10 illustrates the relationship and interactions between the e-Chautauqua engine's workcase components and the log agent components that are in charge of the execution of the event logging mechanism proposed in this paper. As shown in the figure, once one of the workflow clients requests its assigned activity enactment services, then its corresponding workcase component performs the requested services, and makes their corresponding events log information according to the XML-based log message format to be specified in the next section and stores to log databases through the corresponding log agent. The functional structure of the workflow event logging mechanism consists of the following three types of components:

- *Event triggering components* — Requester and Worklist Handler

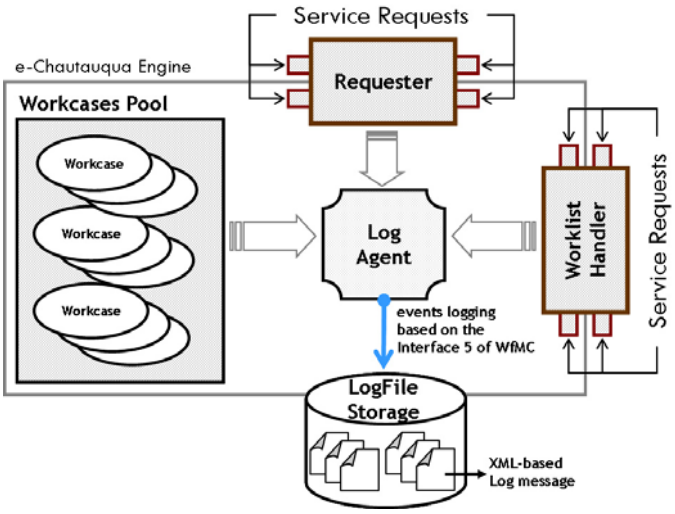

FIGURE 10. e-Chautauqua's events logging mechanism

- *Event formatting components* — Workcase Pool
- *Event logging components* — Log Agent and Log File Storage

The *Requester and Worklist Handler* handle the workflow enactment services requested from workflow clients, and, these services are able to be categorized into three levels of classification—Workcase level class, Running activity level class, and Workitem level class. The *Workcase Pool* has workcase objects, as its residents, each of which composes event log messages according to the service classes after performing the requested services. Finally, the *Log Agent and Log File Storage* take in charge of the responsibility of the event logging mechanism. Once, a log agent receives event logs and then transforms them into XML-based log messages, and store the transformed messages onto the Log File Storage.

4.2. **The implementation details of XWELL.** All events coming from the e-Chautauqua workflow enactment components, such as worklist handler, requester and workcases, would have to be logged on a certain type of storage. These events log information is precisely well-defined in the audit and monitoring functions' standard specifications of WfMC[3]. Moreover, an XML-based formats of audit and monitoring information have been recently released and termed BPAF that stands for Business Process Audit Format by WfMC. However, this section tries to identify and classify all events produced by the engine components, and also to define them by a certain form of XML-based representation formats developed by the author's research group only for the framework proposed in this paper.

4.2.1. *Workflow event log information.* As shortly explained in the previous section, the workcase components, which are taking a role of the event formatting component, compose 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 transmit the formatted event log messages to the event logging components—the log agents. Based on the formatted messages, the log agents form the XML-based event log information. In order to efficiently perform these logging-related jobs, we classify the events into three levels of classes—workcase level event class, running activity level event class, and workitem level event class in implementing the workflow process rediscovery

[3]Workflow Management Coalition, http://www.wfmc.org

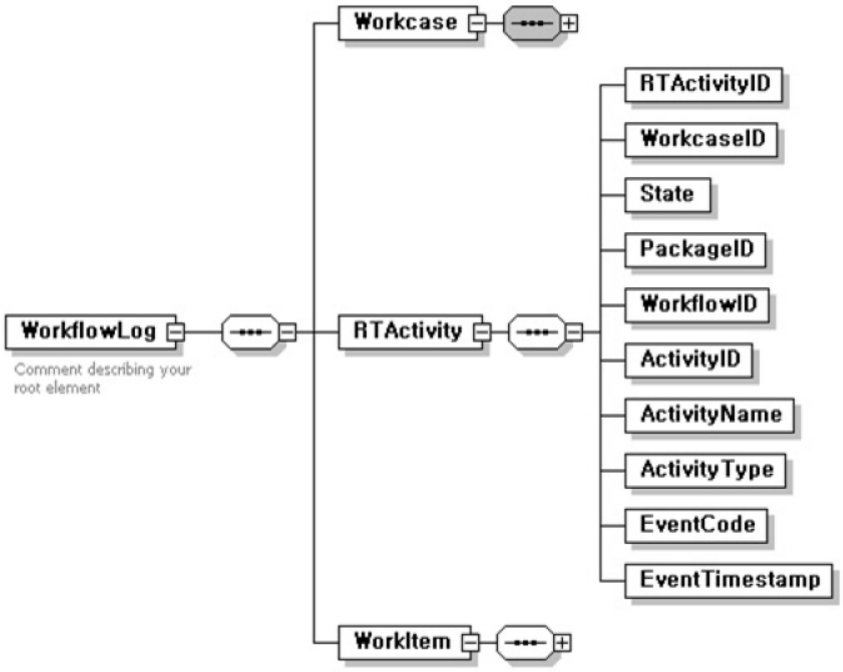

FIGURE 11. Structure of the workflow event log schema

TABLE 4. Workflow event log classes

Event Log Classes	Log Elements to be Tagged in XML
Workcase Class	WorkcaseLog, WorkcaseID, ParentWorkcaseID, WorkcaseName, State, PackageID, WorkflowID, EventCode, EventTimestamp, CreatedTimestamp, StartTimestamp
Running Activity Class	RTActivityLog, WorkcaseID, State, PackageID, WorkflowID, ActivityID, ActivityType, ActivityName, EventCode, EventTimestamp
Workitem Class	WorkitemLog, WorkitemID, PackageID, WorkflowID, WorkcaseID, ActivityID, ActivityName, State, EventCode, EventTimestamp, Performer

framework. Figure 11 shows the event log schema implementing the three levels of the event log classes, and Table 4 enumerates the tagging elements of the event log classes. Also, the detailed event names of the event log classes to be captured and logged by the framework's logging mechanism are simply enumerated as the following:

- *Workcase Level Events*: WMCreatedWorkcase, WMStartedWorkcase, WMChangedWorkcaseState, WMCompletedWorkcase, WMTerminatedWorkcase, WMAbortedWorkcase
- *Running Activity Level Events*: WMChangedActivityInstanceState, WMCompletedActivityInstance, WMTerminatedActivityInstance, WMAbortedActivityInstance
- *Workitem Level Events*: WMAssginedWorkitem, WMTakenWorkitem, WMChangedWorkitemState, WMCompletedWorkitem

4.2.2. *An operational example of the workflow event log.* Figure 12 is a runtime-snapshot of the workflow events logged by the implemented framework's logging mechanism of

FIGURE 12. The implemented workflow event log schema and log-snapshots

the e-Chautauqua workflow management system. The left-hand side of the figure is the XML schema-structure of the workflow event logs, and the right-hand side of the figure shows a snapshot of the sampled XWELL-based workflow event logs, each of which comes from three different types of event log classes, respectively. In order to make a temporal workcase from the logs, the tagged values of WorkcaseID, ActivityID, EventCode, EventTimestamp, and State elements of the workitem class are directly related to form a temporal workcase. Therefore, the σ-Algorithm internally works based on these tagged elements of temporal workcases in the workflow process mining warehouse.

4.3. **Extensions of the implemented framework.** The workflow enactment event log information generated by the implemented rediscovery framework may have some valuable implications on performing a series of advanced techniques not only for improving the quality of workflows but also for discovering useful knowledge. In order to maximize the usability of the valuable logs, it is necessary for the framework to extend its operational functionality and coverage. Conclusively, the framework can be systematically implemented and functionally extended with the following crucial functionalities:

- Workflow runtime status monitoring and statistical reporting functionality
- Workflow runtime recovery functionality
- Workflow knowledge mining and discovery functionality
- Workflow validation functionality

Additionally, the implemented rediscovery framework may affect the techniques and methods of workflow knowledge discovery, too. Here is an example of workflow knowledge discovery issue; It is the case of reachable-path rediscovery problem [5,6] that gives a way to efficiently rediscover the discrepancy between the original workflow process model as it is built and the enacted workflow process (workcase) as it is actually executed. The

discrepancy, as you can easily imagine, is caused by the alternative paths exhibited on the model. The number of alternative paths on the model will effect on the degree of the discrepancy. For example, after rediscovering a workcase from workflow logs, we need to know along which reachable-path the workcase has followed. This might be very useful knowledge for workflow administrators and designers to redesign and re-estimate the original workflow process model after being elapsed a specific amount of period. Conclusively, this statistical runtime information should be very effective and valuable knowledge for redesigning and reengineering the workflow model, and the proposed framework can be extended so as to support this workflow knowledge discovery issue, too.

The implemented rediscovery framework has something to do with those algorithms for discovering a work-sharing human network embedded on a workflow model, and also for quantifying the degree of working-intimacy among humans through the essential notions of the centrality—degree centrality, betweenness centrality, stress centrality and closeness centrality—sharing the enactment of the workflow procedure. Basically, the recent issue of social network discovery is to define a discovered human network from the workflow enactment event logs, to evaluate the degrees of working-intimacies in the discovered human network, and finally to visualize the degrees of working-intimacies among humans through a commercialized tool used to visualizing social networks, based on the centrality analysis result over the discovered human network.

Summarily speaking, the workflow process rediscovery framework proposed in the paper ought to be an essential component for conceiving techniques and methods resolving those issues introduced above. In the near future, the framework will be extended with some discovery techniques of workflow knowledge and social networks.

5. **Related Works.** So far, the research and development works in the workflow literature concerning about the workflow process rediscovery and mining issues have proposed several algorithms like [1-4,8,9,11-15,17,19]. Others have developed workflow process mining systems and tools [2,6]. Particularly, J. Herbsta and D. Karagiannisb in [2] presented results of their experimental evaluation and experiences of the InWoLvE workflow mining system, which is the first industrial application of the workflow process mining techniques. However, almost all of the contributions focus on the development of the basic functionality of workflow process mining techniques. Particularly, W. M. P. van der Aalst's research group, through the papers of [1,9,14,17], proposed the fundamental definition and the use of petrinet-based workflow process mining algorithms to support the design of workflows. They described challenging problems and some workflow process mining approaches and algorithms. Problems, which they stated in [1], include short-loops (one-length loop and two-length loop) problems, invisible task, duplicate task, implicit places, non-free choice, and synchronization of OR-join place problems. However, these problems are based on the Petri-Net modeling approach and focused on the workflow's control flow perspective.

Clarence Ellis's research group newly defined the scope of the workflow mining concept from the view point of workflow systems being "people systems" that must be designed, deployed, and understood within their social and organizational contexts. Thus, they argue in [11,12,19] that there is a need to expand the concept of workflow discovery beyond the process dimension to encompass multidimensional perspective such as social, organizational [22], and informational [20,23] perspectives. As you have seen, this paper proposes the workflow process rediscovery framework based upon the extension version of the σ-*Algorithm* [18], which might be one of the pioneering works pursuing the workflow's multidimensional mining activities. Particularly, this paper showed the feasibility of the framework by implementing the most essential components of the workflow event log formats and language.

Summarily speaking, as a next step of this pioneering work it is needed to investigate the adoptability of the itemsets concepts [21,24] in realizing a certain type of advanced workflow knowledge rediscovery frameworks. Additionally, none of these approaches and algorithms addressed in this paper considers the workflow's change version management [23] activity in rediscovering workflow processes and knowledge. Therefore, adopting these advanced concepts to the workflow rediscovery techniques ought to be the future works of this paper.

6. **Conclusion.** This paper proposes a framework for rediscovering structured workflow processes, and implements the essential components of the framework. Particularly, it shows that the workflow process mining algorithm (σ-*Algorithm*) is able to rediscover structured workflow processes from the temporal workcases filtered from the workflow process execution log. Remind that the σ-*Algorithm* is based on the ICN-based structured workflow process model, which might be the fundamental property differentiating this paper's framework from others'. Additionally, we have seen that the proposed framework is able to correctly handle the three different types of control transitions—sequential, selective and parallel transitions—via an operational example. At the same time, the framework should be improved so as to reasonably rediscover the loop transitions, which should be one of the toughest challenges that the workflow literature conquers in the future. Conclusively, workflow mining methodologies and systems are rapidly growing and coping with a wide diversity of domains in terms of their applications and working environments. Thus, the literature needs various, advanced, and specialized workflow process mining techniques and architectures that are used to eventually give feed-backs to the redesign and reengineering phase of the existing workflow models. We strongly believe that this work might be a pioneering contribution towards improving and advancing workflow rediscovery technology.

Acknowledgement. This work (Grant No. 00047962) was supported by the 2011 Business Grants for Cooperative Research and Development between Industry, Academy, and Research Institutes funded from the Korea Small and Medium Business Administrations.

REFERENCES

[1] W. M. P. van der Aalst et al., Workflow mining: A survey of issues and approaches, *Journal of Data & Knowledge Engineering*, vol.47, no.2, pp.237-267, 2003.

[2] J. Herbsta et al., Workflow mining with InWoLvE, *Journal of Computers in Industry*, vol.53, no.3, 2004.

[3] G. Schimm, Mining exact models of concurrent workflows, *Journal of Computers in Industry*, vol.53, no.3, 2004.

[4] S. S. Pinter et al., Discovering workflow models from activities' lifespans, *Journal of Computers in Industry*, vol.53, no.3, 2004.

[5] K. Kim and C. A. Ellis, Workflow reduction for reachable-path rediscovery in workflow mining, *Series of Studies in Computational Intelligence: Foundations and Novel Approaches in Data Mining*, vol.9, pp.289-310, 2006.

[6] K. Kim, A workflow trace classification mining tool, *International Journal of Computer Science and Network Security*, vol.5, no.11, pp.19-25, 2005.

[7] K. Kim et al., A XML-based workflow event logging mechanism for workflow mining, *Proc. of Int. Conf. on CSA*, 2007.

[8] R. Agrawal et al., Mining process models from workflow logs, *Proc. of Int. Conf. on Extending Database Technology*, 1998.

[9] A. K. A. de Medeiros et al., Process mining: Extending the alpha-algorithm to mine short loops, *BETA Working Paper Series*, 2004.

[10] C. Ellis, Information control nets: A mathematical model of information flow, *ACM Proc. of Conf. on Simulation, Modeling and Measurement of Computer Systems*, pp.225-240, 1979.

[11] C. Ellis et al., Workflow mining: Definitions, techniques, and future directions, *Workflow Handbook 2006*, pp.213-228, 2006.
[12] C. Ellis et al., Beyond workflow mining, *Lecture Notes in Computer Science*, vol.4102, pp.49-64, 2006.
[13] R. Silva, J. Zhang and J. G. Shanahan, Probabilistic workflow mining, *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining*, 2005.
[14] W. M. P. van der Aalst, A. K. A. de Medeiros and A. J. M. M. Weijters, Genetic process mining, *Proc. of Int. Conf. on ATPN*, pp.48-69, 2005.
[15] W. Gaaloul and C. Godart, Mining workflow recovery from event based logs, *Lecture Notes in Computer Science*, vol.3649, pp.169-185, 2005.
[16] R. Liu and A. Kumar, An analysis and taxonomy of unstructured workflows, *Lecture Notes in Computer Science*, vol.3649, pp.268-284, 2005.
[17] W. M. P. van der Aalst et al., Workflow mining: Discovering process models from event logs, *IEEE Transactions on Data & Knowledge Engineering*, vol.16, no.9, pp.1128-1142, 2004.
[18] K. Kim and C. A. Ellis, *σ-Algorithm*: Structured workflow process mining through amalgamating temporal workcases, *Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2007.
[19] A. Rembert, *Automatic Discovery of Workflow Models*, Ph.D. Thesis, University of Colorado at Boulder, 2008.
[20] N. Nakano, A study on modeling and analysis of agent-based simulations with Q-learning, *International Journal of Innovative Computing, Information and Control*, vol.7, no.1, pp.51-60, 2011.
[21] C.-P. Lai, P.-C. Chung and V. S. Tseng, A novel algorithm for mining fuzzy high utility itemsets, *International Journal of Innovative Computing, Information and Control*, vol.6, no.10, pp.4347-4361, 2010.
[22] B. Behdani, Z. Lukszo, A. Adhitya and R. Srinivasan, Agent-based modeling to support operations management in a multi-plant enterprise, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.2873-2884, 2010.
[23] D. Kim, N. Lee, S. Kang, M. Cho and M. Kim, Business process version management based on process change patterns, *International Journal of Innovative Computing, Information and Control*, vol.6, no.2, pp.567-575, 2010.
[24] C.-J. Chu, V. S. Tseng and T. Liang, Mining temporal rare utility itemsets in large databases using relative utility thresholds, *International Journal of Innovative Computing, Information and Control*, vol.4, no.11, pp.2775-2792, 2008.
[25] K. Kim and H. Ahn, An EJB-based very large scale workflow system and its performance measurement, *Proc. of the 6th International Conference on Web-Age Information Management, LNCS*, vol.3739, pp.526-537, 2005.
[26] K.-H. Kim and C. A. Ellis, Performance analytic models and analysis for workflow architectures, *Journal of Information Systems Frontiers*, vol.3, no.3, pp.339-355, 2001.
[27] D. Alonso et al., Exotica/FMQM: A persistent message based architecture for distributed workflow management, *Proc. of the IFIPS Working Conference on Information Systems for Decentralized Organizations*, 1995.
[28] K. Wallnau et al., Toward a distributed mediated architecture for enterprise-wide workflow management, *Proc. of the NSF Workshop on Workflow and Process Automation*, 1996.
[29] J. Weissenfels et al., An overview of the mentor architecture for enterprise wide workflow management, *Proc. of the NSF Workshop on Workflow and Process Automation*, 1996.
[30] Joint Submitters, *Workflow Management Facility*, Revised Submission, OMG Document Number: bom98-06-07, 1998.