Control-path Oriented Workflow Intelligence Analyses*

MINJAE PARK AND KWANGHOON KIM
Collaboration Technology Research Lab.
Department of Computer Science
Kyonggi University
Sawonsi Kyonggido, 442-760, South Korea
E-mail: {mjpark; kwang}@kyonggi.ac.kr

This paper proposes two kinds of control-path oriented workflow knowledge analysis approaches which will be applied to a workflow intelligence and quality improvement framework aiming at the high degree of the workflow traceability and rediscoverability. The framework needs two kinds of algorithms — One is for generating the total sequences of the control-paths from a workflow model, and the other is for rediscovering the runtime enactment history of each control-path out of the total sequences from the corresponding workflow’s execution logs. The proposed approaches have something to do with the former, and each of them include a set of detailed algorithms for analyzing the total sequences of the control-path perspective of a workflow model. Eventually, the analyzed results from the approaches will be effectively used for some rediscovery approaches to rediscover control-path oriented workflow intelligence from the runtime enactment history, which is called workflow events log. Based upon those control-path oriented workflow intelligences, it will be realized for a workflow model not only to be gradually refined, but also to be finally revealed to maximize its quality by repeatedly redesigning, reengineering and/or refining during its whole life-long time period.

Keywords: workflow intelligence and quality, workflow control-path analysis and rediscovery, workflow traceability, workflow knowledge analysis, workflow mining and rediscoverability

1. INTRODUCTION

Until now, it may be fairly said that the research and development efforts in the workflow literature have focused on defining and implementing the basic functionality and services of workflow and business process automation technologies. However, according for those workflow design and automation technologies to swiftly grow and be increasingly used by both traditional and newly-formed web-based enterprises, we need to deal with and attempt to analyze a new and advanced type of requirements and demands concerning workflow intelligence and quality in terms of not only the design-time workflow verification and validation issues but also the runtime workflow execution issues. Furthermore, in order to improve the intelligence of workflows with a high, consistent and predictable quality, especially emphasizing on the workflow traceability and rediscoverability is never enough — there might be several perspectives on them, for instance, the workflows should be correctly designed, their execution should be met their workload requirements, and the (human or automated) process resources should be able

Received September 7, 2005; revised October 16, 2006; accepted September 20, 2007.
Communicated by Tei-Wei Kuo.
* This research was supported by the contents convergence software research center funded by the GRRC program of Gyeonggi Province, Republic of Korea.
to perform the work items in a timely fashion. Out of those perspectives on the workflow traceability and rediscoverability, the main focus of this paper is on the control-path oriented workflow intelligence that ratifies an answer for the question—“Is a workflow model correctly reflecting the control-path aspect of the corresponding real-world business process?”

The most essential issue in the control-path oriented workflow intelligence is to look for efficient workflow execution trace and rediscovery algorithms, the results of which are eventually applied to redesigning and reengineering approaches for maintaining the higher degree of quality on workflow models. Especially, the issue proposed in this paper is much more important because not only the workflows and business processes are becoming massively large-scaled and complicated [5], but also, nevertheless, their life-cycle and recycling (reengineering) periods are becoming swiftly shorter. That is, the quality improvement and refinement works such as redesigning, reengineering, and restructuring of the business processes and workflows should be done after analyzing their behavioral patterns on runtime, and so the workflow trace and rediscovery techniques ought to be the right approaches to explore them from the enactment audit and history information logged and collected from the runtime and diagnosis phases of the workflow procedures. To collect the information, many decisions so need to be made in terms of many perspectives, such as control and data flows [8, 21], work assignment policies and algorithms [18], logging and diagnosis information [14], workcase firing sequences (reachable paths) [12], and the number of workcases performed and managed by various users and administrators. These decisions are done based upon the information and knowledge collected not only in the runtime-diagnosis phases but also in the (re)design phase. So far, several approaches and algorithms [8, 18] conducting the work of the runtime-diagnosis phases have been proposed in the literature. While one the other hand, there is a few approach [12] collecting knowledge during the design phases. So, this paper proposes two approaches to generate a series of control-paths intelligence of a workflow model during the design phases by analyzing control-paths of the underlying workflow model. Additionally, we strongly believe that those analyzed control-paths intelligence will be effectively used to rediscovering control-path oriented intelligence from the workflow logs gathered in the model’s runtime execution history.

The remainder of this paper is organized as follows. The next section clearly states the issue of the control-path oriented workflow intelligence. And in the next, we work out the framework to define the workflow model’s control-path analysis methods and its execution trace and rediscovery methods in a formal approach. Based on the methods, we implement their algorithms to rediscover and classify control-paths of a large class of workcases, and describes about the implications of the workflow intelligence and quality framework. Finally, we finalize with a brief description of its related works and conclusions including future works.

2. THE PROBLEM SCOPE: THE QUALITY OF WORKFLOW MODEL

In this paper, our emphasis is placed on the quality of workflow model [3], and especially on the control-path oriented workflow intelligence that aims at improving the quality of the corresponding workflow model. Then, what is the quality of a workflow
model? And how can we evaluate the quality of a workflow model? So, in this paper, we try to answer for those questions through the concept of the control-path oriented intelligence. We would insist that the quality of a workflow model be defined by the degree of the discrepancy between the defined workflow model as it is built on modeling time and a set of activity firing sequences (classified by control-paths), each of which is called workcase, as it is actually executed on running time. The discrepancy, as you can easily imagine, is caused by disjunctive and conjunctive control-paths presented on the model. The number of disjunctive control-paths on the model will effect on the number of mutually exclusive activity firing sequences. Also, the number of activities associated with a conjunctive path will effect on the number of mutually inclusive activity firing sequences.

Fig. 1. The control-path oriented workflow intelligence analysis and rediscovery.

Fig. 1 shows the control-path oriented workflow intelligence of a workflow model. The left-hand side of the figure shows the analyzed control-path oriented intelligence built on modeling time, and the right-hand side of the figure represents the rediscovered control-path oriented intelligence generated from running time traces and execution logs. The workflow model in the figure has three mutually exclusive activity firing sequences, and the rightmost one of which possibly generates six mutually inclusive activity firing sequences. For example, after fetching a workcase from workflow logs, we need to know along which control-path (or firing sequence of activities) the workcase has followed or enacted. This might be very useful knowledge for workflow administrators and designers to redesign and re-estimate the quality of the workflow model after being elapsed a specific amount of period. Fig. 2 shows an example of the workflow quality based upon the concept of control-path oriented intelligence, which may be possibly measured after being elapsed a certain amount of time. Based upon the quality measurements, the workflow model can be qualitatively improved by separating the control-path having the smallest number of workcases from the remainder control-paths through the quality improvement and refinement works.

As a matter of fact, this paper proposes a feasible framework to handle out the workflow quality issue through control-path oriented intelligence as shown in Fig. 3. The framework consists of the analysis part and the rediscovery part for the control-path
oriented workflow intelligence. The analysis part (the upper part of the figure) takes in charge of analyzing control-path oriented workflow intelligence by generating a set of control-paths from a workflow model. The rediscovery part (the lower part of the figure) is of rediscovering or mining the corresponding control-paths’ instances (workcases) from the workflow warehouse built from workflow logs. The main research scope of this paper is on the upper part of the framework. That is, this paper’s goals are to define what the control-path oriented workflow intelligence is, and to develop some algorithms that is able to automatically extract the intelligence out of workflow models. In the next section, we describe those approaches and algorithms for the analysis part of the framework.
3. ANALYSIS APPROACHES FOR CONTROL-PATH ORIENTED WORKFLOW INTELLIGENCE

In this section, we give the complete descriptions of formal approaches to analyze control-path oriented intelligence from a workflow procedure modeled by the information control net [2, 3, 17]. We have contrived two analysis approaches – one is generic approach, the other is transformation-driven approach. The generic analysis approach is a straightforward way because it analyzes and produces a set of control-paths by traversing the targeted workflow model. The transformation-driven analysis approach is a sophisticated way, but an efficient way, because it transforms the targeted workflow model into a control-path decision tree. This approach is especially contrived for easily classifying the targeted workflow model’s workcases according to its control-path intelligence. The following subsections devote to details and formalities of the control-path oriented workflow intelligence analysis approaches.

3.1 Information Control Net for Workflow Model

The control-path oriented workflow intelligence and its analysis approaches are based on the Information Control Net (ICN) [17] that is a typical workflow modeling methodology. We formally describe the ICN-based workflow model [3]. An ICN captures these notations of workflow procedures, activities, precedence, and repositories. A workflow procedure is a predefined set of work steps, called activities, and a partial ordering of these activities. Activities can be related to each other by conjunctive logic (after activity A, do activities B and C) or by disjunctive logic (after activity A, do activity B or C) with predicates attached. An activity is either a compound activity containing another procedure, or a basic unit of work called an elementary activity. An elementary activity can be executed in one of three modes: manual, automatic, or hybrid. Typically one or more participants are associated with each activity via roles. A role is a named designator for one or more participants which conveniently acts as the basis for partitioning of work skills, access controls, execution controls, and authority/responsibility. An actor is a person, program, or entity that can fulfill roles to execute, to be responsible for, or to be associated in some way with activities and procedures.

Fig. 4 is to represent an example of ICN-based workflow model (the hiring workflow model [6]) by its graphical notations. In the graphical notation of ICN, circles represent activities, arcs (which may have transition conditions such as reject and accept) represent the precedence partial order, hollow dots represent or-split and or-join in a pair of nodes, filled dots represent and-split and and-join in a pair of nodes, and small boxes represent relevant data. So, three parts of the figure graphically represent control-flow/activity-role assignment, relevant data-flow, and role-actor assignment, respectively. Also, the following Definition 1 is the formal description of the information control net.

Definition 1 Information Control Net (ICN) of a workflow model. A basic ICN is 8-tuple $\Gamma = (\delta, \gamma, \lambda, \varepsilon, \pi, \kappa, I, O)$ over a set of $A$ activities (including a set of group activities), a set $T$ of transition conditions, a set $R$ of repositories, a set $G$ of invoked application programs, a set $P$ of roles, and a set $C$ of actors (including a set of actor groups), where
Fig. 4. The ICN-based hiring workflow model.

- $I$ is a finite set of initial input repositories, assumed to be loaded with information by some external process before execution of the ICN;
- $O$ is a finite set of final output repositories, perhaps containing information used by some external process after execution of the ICN;
- $\delta = \delta_i \cup \delta_o$ where, $\delta_i: A \rightarrow \wp(\alpha \in A)$ is a multi-valued mapping function of an activity to its set of (immediate) successors, and $\delta_o: A \rightarrow \wp(\alpha \in A)$ is a multi-valued mapping function of an activity to its set of (immediate) predecessors;
- $\gamma = \gamma_i \cup \gamma_o$ where $\gamma_i: R \rightarrow \wp(\alpha \in A)$ is a multi-valued mapping function of an activity to its set of output repositories, and $\gamma_o: R \rightarrow \wp(\alpha \in A)$ is a multi-valued mapping function of an activity to its set of input repositories;
- $\lambda = \lambda_o \cup \lambda_p$ where $\lambda_o: G \rightarrow \wp(\alpha \in A)$ is a single-valued mapping function of an activity to its invoked application program, and $\lambda_p: A \rightarrow \wp(\tau \in G)$ is a multi-valued mapping function of an invoked application program to its set of associated activities;
- $\epsilon = \epsilon_o \cup \epsilon_p$ where $\epsilon_o: P \rightarrow \wp(\alpha \in A)$ is a single-valued mapping function of an activity to one of the roles, and $\epsilon_p: A \rightarrow \wp(\eta \in P)$ is a multi-valued mapping function of a role to its sets of associated activities;
- $\pi = \pi_o \cup \pi_p$ where $\pi_c: C \rightarrow \wp(\eta \in P)$ is a multi-valued mapping function of a role to its sets of associated actors, and $\pi_p: P \rightarrow \wp(o \in C)$ is a multi-valued mapping function of an actor to its sets of associated roles;
- $\kappa = \kappa_i \cup \kappa_o$ where $\kappa_c(\alpha)$: sets of control-transition conditions, $T$, on each arc, $\delta_i(\alpha)$, $\alpha \in A$; and $\kappa_o(\alpha)$: sets of control-transition conditions, $T$, on each arc, $\delta_o(\alpha)$, $\alpha \in A$; where the set $T = \{\text{default}, \text{or (conditions)}, \text{(conditions)})$.
3.2 Generic Analysis Approach

The generic approach for generating a set of control-path oriented intelligence from an ICN-based workflow model is a straightforward approach as one can expect from the term, generic. That is, as shown in Fig. 5, it automatically produces a set of control-paths by traversing the targeted workflow model. The control-path oriented intelligence is formally defined through the Control-path Net of Definition 2, and it is automatically produced by the following algorithm, PROCEDURE TRAVERSE().

![Fig. 5. The generic Approach for Analyzing control-path intelligence.](image)

**Definition 2**  *Control-path Net* of a workflow model (ICN). Let $W$ be a $CpN$, a control-path net, that is formally defined as $CpN = (\varnothing, \kappa, I, O)$ over a set of activities, $A^p$, and a set of transition-conditions, $T^p$, where

- $\varnothing = \varnothing_i \cup \varnothing_o$
  - where, $\varnothing_i: A^p \rightarrow \wp(\alpha \in A^p)$ is a multi-valued mapping of an activity to its set of (immediate) successors, and $\varnothing_o: A^p \rightarrow \wp(\alpha \in A^p)$ 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, $\tau \in T^p$, on each arc, $(\beta_i(\alpha), \alpha)$; and $\beta_o(\alpha)$: a set of control transition conditions, $\tau \in T^p$, on each arc, $(\alpha, \beta_o(\alpha))$, where $\alpha \in A^p$;
- $I$ is a finite set of initial input repositories of the corresponding ICN;
- $O$ is a finite set of final output repositories of the corresponding ICN.

**The Generic Analysis Algorithm**  PROCEDURE TRAVERSE():

**Input:** An Information Control Net (ICN), $\Gamma = (\delta, \gamma, \lambda, \pi, \kappa, I, O)$;
**Output:** A Set of Control-path Nets ($CpNs$), $\forall W = (\varnothing, \kappa, I, O)$;
**Initialize:** $CpN \leftarrow \{\varnothing\}$; /* The empty net of $CpN$. */
**PROCEDURE TRAVERSE(In $s \leftarrow \{\alpha\}, CpN$) */ Recursive Call. */
**BEGIN**
\[ v \leftarrow s; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{v\}; \]
\[
\text{WHILE } \left( (u \leftarrow \delta(u)) \neq \{\alpha_p\} \right) \text{ DO}
\]
\[
\text{SWITCH } (\text{What type of the activity, } u, \text{ is?}) \text{ DO}
\]
\[
\text{Case \textquote{\textsc{serial-type activity}}:}
\]
\[
w \leftarrow u; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{w\};
\]
\[
CpN.A_{\alpha}(v) \leftarrow w; \ CpN.A_{\alpha}(w) \leftarrow v;
\]
\[
CpN.\beta(v) \leftarrow \kappa(s); \ CpN.\beta(v) \leftarrow \kappa(s);
\]
\[
b \leftarrow \text{break};
\]
\[
\text{Case \textquote{\textsc{conjunctive-type (AND-split) activity}}:}
\]
\[
w \leftarrow u; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{w\};
\]
\[
CpN.A_{\alpha}(v) \leftarrow w; \ CpN.A_{\alpha}(w) \leftarrow v;
\]
\[
CpN.\beta(v) \leftarrow \kappa(s); \ CpN.\beta(v) \leftarrow \kappa(s);
\]
\[
\text{FOR } (\text{each of } \forall a \in \delta_u(u)) \text{ DO}
\]
\[
x \leftarrow a; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{x\};
\]
\[
CpN.A_{\alpha}(w) \leftarrow x; \ CpN.A_{\alpha}(x) \leftarrow w;
\]
\[
CpN.\beta(w) \leftarrow \kappa(a); \ CpN.\beta(w) \leftarrow \kappa(a);
\]
\[
\text{END FOR}
\]
\[
\text{END FOR}
\]
\[
\text{Case \textquote{\textsc{disjunctive-type (OR-split) activity}}:}
\]
\[
w \leftarrow u; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{w\};
\]
\[
CpN.A_{\alpha}(v) \leftarrow w; \ CpN.A_{\alpha}(w) \leftarrow v;
\]
\[
CpN.\beta(v) \leftarrow \kappa(s); \ CpN.\beta(v) \leftarrow \kappa(s);
\]
\[
\text{FOR } (\text{each of } \forall a \in \delta_u(u)) \text{ DO}
\]
\[
\text{Call PROCEDURE TRAVERSE(In } s \leftarrow a, \ CpN);\]
\[
\text{END FOR}
\]
\[
b \leftarrow \text{break};
\]
\[
\text{Default: } /* \text{OR-join activity or AND-join activity */}
\]
\[
w \leftarrow u; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{w\};
\]
\[
CpN.A_{\alpha}(v) \leftarrow w; \ CpN.A_{\alpha}(w) \leftarrow v;
\]
\[
CpN.\beta(v) \leftarrow \kappa(s); \ CpN.\beta(v) \leftarrow \kappa(s);
\]
\[
b \leftarrow \text{break};
\]
\[
\text{END SWITCH}
\]
\[
s \leftarrow u; \ v \leftarrow w;
\]
\[
\text{END WHILE}
\]
\[
w \leftarrow u; \ CpN.A^p \leftarrow \ CpN.A^p \cup \{w\}; /* u is equal to } \alpha_p * /
\]
\[
CpN.A_{\alpha}(v) \leftarrow w; \ CpN.A_{\alpha}(w) \leftarrow v;
\]
\[
CpN.\beta(v) \leftarrow \kappa(s); \ CpN.\beta(v) \leftarrow \kappa(s);
\]
\[
\text{PRINTOUT } CpN
\]
\[
\text{END PROCEDURE}
\]

The time complexity of the generic analysis algorithm is \(O(n)\), where \(n\) is the number of activities in an ICN, because each traverse() is recursively traversing each activity only once. Therefore, the overall time complexity is \(O(n)\).
3.3 Transformation-driven Analysis Approach

This section describes a transformation-driven analysis approach for analyzing the control-path oriented intelligence, which consists of a serial of formal approaches from analyzing activity dependencies to generating a control-path intelligence decision tree, and their related algorithms, which is finally used for rediscovering workcases of control-path oriented intelligence. This approach is quite different from the previous generic approach in terms of the operational efficiency producing the control-path knowledge from the execution logs. (Note that the analyzed results which are the control-path knowledge of the underlined workflow model, will be used by a rediscovery algorithm which is operating based on the analyzed results.) The analyzed results from the generic algorithm are a set of control-paths, each of which consists of all activities that are involved in the control-path. While on the other, the analyzed results generated by the approach to be presented in this section are a set of activities, each of which is a representative (or key) activity of its control-path. Therefore, the rediscovery algorithm is able to make a decision, to which control-path a certain workcase is belonged, just by referring the key activity of each control-path. The approach, as illustrated in Fig. 6, has two kinds of model transformations. The first transformation is from an Information Control Net (ICN) to an activity dependent net performed through the activity dependency analysis algorithm, and the second is from the activity dependent net to a control-path oriented intelligence decision tree. These transformations are performed in the modeling time, and finally used in classifying workcases’ control-paths from workflow log. This approach firstly steps forward from analyzing control dependencies among activities in an ICN-based workflow model.

Based upon these control dependencies, the approach generates an activity dependent net by applying the concepts of walk and dominance operations [1]. On the activity dependent net, it is possible to filter out a control-path oriented intelligence decision tree model by using the concepts of immediate backward dominator and immediate forward dominator [1]. Finally, the decision tree model serves as a decision tree induction algorithm (generating reduced activity set or minimal activity set) in order to decide along which control-path a workcase follows. In the next two subsections, we formally describe about the activity dependent net and the decision tree model of the approach.
The Activity Dependency Analysis  This subsection describes the formal definitions of activity dependent net, and an algorithm that constructs an activity dependent net from the formal specification of an ICN. The activity dependent net is mainly concerned about the control-flow perspective in a workflow procedure. In particular, it is used to model the effect of conditional branches (or-split and or-join) and parallel branches (and-split and and-join) on the behavior of workflow procedure. The activity dependent net is constructed out of the set \( \delta \) (control flow part) in the formal notation of ICN-based workflow model. Note that the notations, operations and their meanings, which are closely related with the dependency analysis, are same in [12]. So, we do not described in this paper.

Activity Dependent Net of Definition 3  Based upon the primitive operations (walk and dominance) [15], it is possible to generate a set of activity control dependent relationships being embedded on a workflow model. We can automatically construct an activity dependent net of the workflow model by an algorithm.

Definition 3  Workflow Dependent Net of an ICN. A workflow dependent net is formally defined as \( \Omega = (\phi, \xi, I, O) \) over a set \( A \) of activities and a set \( T \) of transition conditions, where

- \( \phi = \phi_1 \cup \phi_o \)
  where \( \phi_a : A \to \phi(A) \) is a multi-valued mapping of an activity to a set of activities that is control-dependent or strongly or multiply control-dependent on the activity, and \( \phi : A \to \phi(A) \) is a single-valued mapping of an activity to that the activity is control-dependent or strongly or multiply control-dependent;
- \( \xi = \xi_i \cup \xi_o \)
  where \( \xi_i \) : a set of control transition conditions, \( \tau \in T \), on each arc, \( (\phi(\alpha), \alpha) \); and \( \xi_o \) : a set of control transition conditions, \( \tau \in T \), on each arc, \( (\alpha, \phi(\alpha)) \), where \( \alpha \in A \);
- \( I \) is a finite set of initial input repositories;
- \( O \) is a finite set of final output repositories.

In mapping the activity dependent diagram into its formal definition, a circle represents an activity node, a solid arrow represents a control-dependency between two associated activities, and a control-transition condition is positioned on the solid arrow. Additionally, the activity dependent net is extensible to accommodate the concept of compound workflow models such as subworkflows, nested workflows, and chained workflows. Next, how can we build an activity dependent net from an information control net? We need an algorithm for this. By using those operations such as walk,dominance and control-dependency, we construct the algorithm that generates the activity dependent net as shown in the middle part of Fig. 6, which is the graphical representation of the activity dependent net.

The Activity Dependent Net Construction Algorithm

**PROCEDURE DEPENDENCE()**:

**Input**: An ICN, \( \Gamma = (\delta, \gamma, \varepsilon, \pi, \kappa, I, O) \);

**Output**: A Workflow Dependent Net, \( \Omega = (\phi, \xi) \);

**Initialize**: \( T \leftarrow \{ \alpha_i \} \); /* \( u, T \) are global */
The Control-path Oriented Intelligence Decision-Tree

In this subsection, we formally define the Control-path Oriented Intelligence decision tree model, and derive an algorithm constructing itself from an activity dependent net. In order to construct the model, it is necessary to extend the domination-relationship operations [1] (such as ifd and ibd) so as to incorporate the concept of dependency type. In the workflow dependent net, we treat every node in a net as a unified type - activity. However, in the decision tree model, it is necessary for the nodes to be classified into activity-type with the immediate backward domination, conjunctive-logic-type (and-split), and disjunctive-logic-type (or-split), which are playing an important role in composing the model. Based upon these operations and classes, the types of dependency are defined as following:

Definition 4 Types of Dependency in a workflow dependent net model (WDN). Let \( \Omega \) be a WDN, \( \Omega = (\phi, \delta, I, O) \) over a set of activities, \( A \), and a set of transition-conditions, \( T \). We can define the following types of dependency:

- There exists an ibd-type dependency between two activities, \( v \) and \( u \),
  where \( v \in \phi(u) \land u \in \phi(v) \land v \neq u \) in WDN, which is denoted as ibdtd\( v \) and gives an

---

The time complexity of the activity dependent net construction algorithm is \( O(n^2) \), where \( n \) is the number of activities in an ICN. The functions for deciding both the strongly forward-domination relation [1] and the multiply forward-domination relation [1] between two activities can be computed in \( O(n) \) [1], and the recursive procedure itself can be computed in \( O(n) \). Therefore, the overall time complexity is \( O(n^2) \).
activity, \( u \), that is the immediate backward dominator of \( v \) occurring on every \( v - a \) walk in an ICN.

- There exists a conjunction-type dependency between two activities, \( v \) and \( u \), where \( v \in \phi(u) \land u \in \phi(v) \land v \neq u \) in WDN, which is denoted as \( \text{ctd}(v) \) and gives an activity, \( u \), that is a ‘and-split’ activity.

- There exists a disjunction-type dependency between two activities, \( v \) and \( u \), where \( v \in \phi(u) \land u \in \phi(v) \land v \neq u \) in WDN, which is denoted as \( \text{dtd}(v) \) and gives an activity, \( u \), that is a ‘or-split’ activity.

**Definition 5**  
Control-path Oriented Intelligence Decision Tree of an activity dependent net model. Let \( M \) be a decision tree, that is formally defined as \( M = (\chi, \vartheta, I, O) \) over a set of activities, \( A \), and a set of transition-conditions, \( T \), where

- \( \chi = \chi_i \cup \chi_o \)
  - \( \chi : A \rightarrow \mathcal{P}(A) \) is a multi-valued mapping of an activity to another set of activities, each member of which has one of the types of dependency, such as ibd-type, conjunctive-type, or disjunctive-type dependency, and \( \chi_i : A \rightarrow \mathcal{P}(A) \) is a single-valued mapping of an activity to another activity that is one of the members in \( \{a_i\} \), or-split, and-split;
- \( \vartheta = \vartheta_i \cup \vartheta_o \)
  - \( \vartheta : \mathcal{P}(T) \) is a set of control transition conditions, \( \tau \in T \), on each arc, \((\phi(\alpha), \alpha)\); and \( \vartheta_o : \mathcal{P}(T) \) is a set of control transition conditions, \( \tau \in T \), on each arc, \((\alpha, \phi_o(\alpha))\), where \( \alpha \in A \);
- \( I \) is a finite set of initial input repositories;
- \( O \) is a finite set of final output repositories.

In mapping a decision tree diagram into its formal definition, solid directed edge coming into a node correspond to \( \chi_i \), and solid directed edge going out of a node correspond to \( \chi_o \). A decision tree model is formally constructed from an activity dependent net through the following algorithm. In Fig. 6, the right-hand side is to represent a decision tree model extracted from the activity dependent net by the decision tree induction algorithm.

The Control-path Oriented Intelligence Construction Algorithm

**PROCEDURE DECISION():**

**Input:** A Workflow Dependent Net, \( \Omega = (\phi, \xi, I, O) \);

**Output:** A Minimal Workflow Net, \( M = (\chi, \vartheta, I, O) \);

**Initialize:** \( T \leftarrow \{\emptyset\} \); /* \( T \) are global */

**PROCEDURE DECISION(In \( s \leftarrow \{a_i\} \)) /* Recursive procedure */**

**BEGIN**

\( v \leftarrow s; \chi(v) \leftarrow \phi(v); \chi_o(v) \leftarrow \{\emptyset\}; T \leftarrow T \cup \{v\}; \)

\( O \leftarrow \phi_o(v); \)

**FOR** \( (\forall u \in O) \)**

**SWITCH** (What type of dependency between \( v \) and \( u \) is ?) **DO**

**Case ‘ibd-type dependency’:**

\( \chi_o(v) \leftarrow u; \chi_i(u) \leftarrow v; \)

\( \vartheta_i(u) \leftarrow \xi_o(u); \vartheta_o(u) \leftarrow \xi_i(u); \)**
\[
T \leftarrow T \cup \{v\}; \\
\text{break}; \\
\text{Case 'conjunctive-type dependency':} \\
\chi_o(v) \leftarrow u; \quad \chi_i(v) \leftarrow v; \\
\vartheta_o(v) \leftarrow \xi_o(v); \quad \vartheta_i(u) \leftarrow \xi_i(u); \\
T \leftarrow T \cup \{u\}; \\
\text{Call PROCEDURE DECISION(In } s \leftarrow u); \\
\text{break;}
\]

\[
\text{Case 'disjunctive-type dependency':} \\
\chi_o(v) \leftarrow u; \quad \chi_i(v) \leftarrow v; \\
\vartheta_o(v) \leftarrow \xi_o(v); \quad \vartheta_i(u) \leftarrow \xi_i(u); \\
T \leftarrow T \cup \{u\}; \\
\text{Call PROCEDURE DECISION(In } s \leftarrow u); \\
\text{break;}
\]

\[
\text{Default:} \\
T \leftarrow T \cup \{u\}; \\
\text{break;}
\]

\text{END SWITCH}

\text{END FOR}

\[
\text{IF } (x, y \in \chi_o(v)) \land (x \neq y) \land (\vartheta_i(x) = \vartheta_i(y)) \land (x = \text{ibdtp}(v)) \text{ DO} \\
\text{Then do} \\
\text{Eliminate } x \text{ (the idb-type dependencyy)} \\
\text{from the minimal workflow net;}
\]

\text{end;}

\text{END IF}

\text{END PROCEDURE}

The Control-path oriented intelligence construction algorithm for an activity dependent net can be computed in \(O(n)\), where \(n\) is the number of activities in the set, \(A\). Because the statements in the recursive procedure are executed \(n\) times that is exactly same to the number of activities, and the time needed for deciding the immediate backward dominator (ibd-type dependency) \([1]\) is \(O(1)\) \([1]\). Therefore, the time complexity of the algorithm is \(O(n)\).

In summary, the control-path oriented intelligence decision tree model, which is finally generating the minimal activity set consisting of decision activity sets, will play a very important role in control-path oriented intelligence analysis and rediscovery framework as a control-path decision tree induction technique. Based on this minimal activity set, we can easily and efficiently rediscover the number of workcases associated with each of the control-paths making up the workflow intelligence.

4. RELATED WORKS

In recent, workflow (business process) and its related technologies have been constantly deployed and so gradually hot-issued in the IT arena. This atmosphere booming workflows and business processes modeling and reengineering is becoming a catalyst for
triggering emergence of the concept of workflow mining that collects data at runtime in order to support workflow design and analysis for redesigning and reengineering workflows and business processes. Especially real workflow models going with e-Commerce, ERP (Enterprise Resource Planning), and CRM (Customer Relationship Management) are getting larger and more complex in their behavioral structures. These large-scaling movements trigger another big changes in workflow administration (the responsibility of analysis) and monitoring (the responsibility of rediscovery) functionality that has been featured and embedded in the workflow build-time and run-time functionality of the traditional workflow systems. In other words, there have been prevalent research and development trends in the workflow literature workflow mining techniques [7, 10-12, 15, 16, 18-21] and systems [9, 13] that collect runtime data into workflow logs [14], and filter out information and knowledge from the workflow logs gathered by the administration and monitoring features of workflow management system.

However, almost all of the workflow mining techniques have been mainly focusing on the process rediscovery issues as efficient redesigning and reengineering approaches. That is, BPx, such as business process redesign, reengineering, and restructuring, needs to be done frequently and even dynamically as well. In order to perform the BPx efficiently and effectively, they should consider its enactment audit and history information logged and collected from the runtime and diagnosis phases of the workflow and business process management system, which means that these decisions made by the previous techniques have been done based upon the information and knowledge collected only in the runtime-diagnosis phases. On the other hand, in this paper, our emphasis is on the process analysis issues. In a little more detailed idea, this paper gives a fundamental way to efficiently rediscover the discrepancy between the original workflow process model as it is built and the enacted workflow process (which is called from now 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. Conclusively speaking, the algorithms proposed in this paper can be applied to decide the discrepancy degree of the underlining workflow model. Also, the analysis results will be used by the runtime-diagnosis algorithms so as to efficiently rediscovery the control-path oriented workflow intelligence from the workflow execution logs.

5. CONCLUSIONS

So far, this paper has introduced the concept of control-path oriented workflow intelligence and it has also described the formal analysis approaches, and its related algorithms. Particularly, in this paper we newly contrived the control-path oriented workflow intelligence analysis and rediscovery framework, and proposed a serial of feasible solutions for the framework, as well. The framework eventually gives us higher-level of efficiency in improving the quality of workflow models. Conclusively, the analysis approaches become a sort of control-path oriented intelligence decision induction techniques in workflow intelligence analysis and rediscovery systems. In recent, the literature needs various, advanced, and specialized workflow mining techniques and architectures that are used for finally feed-backing their analysis results to the redesign and reengi-
neering phase of the existing workflow and business process models. We strongly believe that this work might be one of those impeccable attempts and pioneering contributions for improving and advancing the workflow intelligence mining technology.

REFERENCES


Minjae Park received the B.S. and M.S. degrees in Computer Science from Kyonggi University, Korea, in 2004 and 2006, respectively, and is currently a Ph.D. candidate in Kyonggi University. His research interests are workflow/process mining, RBAC and RTE.

Kwanghoon Kim is an Associate Professor of Computer Science Department and Director of the Collaboration Technology Research Laboratory at Kyonggi University, South Korea. At Kyonggi, he is involved in research and teaching of workflow, groupware, coordination theory, computer networks, software architectures, and database systems. And, he is in charge of the director of the contents convergence software research center that has been launched at just summer of 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 a 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, a country-chair(Korea) and a 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, data warehousing and mining, software architecture modeling and simulation, e-commerce, and computer networks.