Expert Systems with Applications 26 (2004) 301–310

Expert Systems with Applications

www.elsevier.com/locate/eswa

Discovering role-relevant process-views for disseminating process knowledge

Minxin Shen, Duen-Ren Liu*

Institute of Information Management, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan,

Abstract

Workflow technology supports the management of organizational process knowledge. However, workers (representing organizational roles) cannot easily obtain a global view of a complex and large workflow. This work proposes a process-view based dissemination of process knowledge to address this problem. A process-view is an abstracted process derived from a physical process, and can provide adaptable task granularity. The relationships among tasks, roles and operations, as specified in role-based access control systems, are used to evaluate the degrees of relevance between roles and tasks. Next, a novel algorithm is proposed to generate automatically role-relevant process-views based on degrees of relevance. Knowledge management systems can thus disseminate abstracted process knowledge through process-views for those roles in a complex workflow.

© 2003 Elsevier Ltd. All rights reserved.

Keywords: Process knowledge; Knowledge management; Knowledge dissemination; Workflow management

1. Introduction

Knowledge management (KM) helps organizations to utilize their intellectual capital to gain sustainable competitive advantages (Davenport & Prusak, 1998; Edvinsson & Malone, 1997; Liao, 2002). An organization has various types of knowledge, including for example tacit knowledge, explicit knowledge, and cultural knowledge. Process knowledge refers to knowledge of business processes. Knowledge Management Systems (KMS) support KM activities by integrating information and communication technologies. As an effective process management tool, workflow management systems (WfMS) allow a business to analyze, simulate, design, enact, control and monitor general business processes (Georgakopoulos, Hornick, & Sheth, 1995; Leymann & Altenhuber, 1994). Therefore, managing process knowledge through WfMS or workflow components of KMS is straightforward.

Appropriate knowledge representation is crucial for establishing a successful KMS. Despite notational differences, activity-based methodologies (Georgakopoulos et al., 1995) are extensively used process modeling techniques for representing process knowledge. A typical activity-based approach designs a workflow using a top-down decomposition procedure. This stepwise refinement allows knowledge engineers or process modelers to define a process more easily and completely than do one-step approaches.

In practice, workflow participants possess different needs and types of authority when obtaining information on business processes. For example, a high-level manager may need aggregated information of a process, and a marketing manager may not have the authority and need to know each specific step of production flow. However, the activity-based model cannot flexibly alter task granularity of a process based on role characteristics. Therefore, WfMS may not provide each organizational level and unit with an appropriate view of that process. This work aims to develop a method that can disseminate process knowledge at customized granularity to workers.

Delivering relevant and necessary documents to workers in order to accomplish their tasks in a workflow environment has been addressed by (Abecker, Bernardi, Maus, Sintek, & Wenzel, 2000; Staaba & Schnurr, 2000). However, these studies suggested task-specific information, rather than process-oriented knowledge, such as of the progress status of an entire workflow. Our previous study (Liu & Shen, 2003) described a process-view model that

* Corresponding author. Tel: +886-35712121; fax: +886-35723792.
E-mail address: dliu@iim.nctu.edu.tw (D.-R. Liu).

0957-4174/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2003.09.002

enhances the ability of conventional activity-based process models to abstract processes, and can present a process definition at adaptable task granularity. A process-view, i.e. a virtual process, is abstracted from an actual process. According to the requirements of distinct organizational roles, a process modeler can design various process-views, providing the process information suitable for each participant. However, the preliminary process-view model requires process modelers (domain experts) to define various process-views for distinct roles. Process-view design is complex and time-consuming. Therefore, this work proposes a novel approach to assist the discovery of role-relevant process-views and applies the results to disseminate process knowledge.

This work first presents a quantitative method for evaluating the degrees of relevance between tasks and organizational roles. Different roles may perform different workflow operations on tasks, under the restriction of permission rules in role-based access control systems (Ferraiolo, Sandhu, Gavrila, Kuhn, & Chandramouli, 2001). Thus, permission rules, which prescribe the authorization relationships among roles, tasks and operations, and audit logs of task execution are utilized to measure the degrees of relevance between roles and tasks. Based on the relevance degrees and the granular threshold, novel algorithms are proposed herein to derive role-relevant process-views. Process designers or workers specify the granular threshold to control the granularity of generated process-views. Thus, during workflow enactment, KMS or WfMS can disseminate process knowledge at a suitable granularity to participants; that is, more relevant parts are presented with a finer resolution and less relevant parts are presented more coarsely.

The rest of this paper is organized as follows. Section 2 presents the process-view based dissemination of process knowledge. Formal definitions and the means of constructing an order-preserving process-view, as presented in (Liu & Shen, 2003), are also summarized. Next, Section 3 presents the procedure for evaluating relevance degrees between roles and tasks. The algorithms for discovering role-relevant process-views are also presented. Section 4 then discusses and compares the process-view based knowledge dissemination with related work. Section 5 draws conclusions.

2. Dissemination of process knowledge

A process-view, an abstracted process derived from a physical process, reveals abstracted process information. Process designers can define various role-relevant process-views, according to the characteristics of participating roles, to achieve different levels of information concealment. The following presents process-view based dissemination of process knowledge, and then formally defines base processes, process-views and order-preserving process-views.

2.1. Process-view based knowledge dissemination

A process that may have multiple process-views is referred to herein as a base process. A process-view is generated from either base processes or other process-views and is considered a virtual process. From the users' perspective, a process-view resembles a typical process that consists of activities (tasks) and dependencies although it is an abstracted form of an implemented process.

Fig. 1 illustrates process-view based dissemination of process knowledge. Assume that the base process shown in Fig. 1 is a manufacturing process. Marketers do not need to know every step in the process, although they must know the progress of order fulfillment to serve their customers. A process modeler can design an appropriate process-view for the marketing department as follows: a_1, a_2, and a_3 are mapped into va_1; a_4 and a_5 are mapped into va_2; a_6 and a_7 are mapped into va_3 (i.e. $va_1 = \{a_1, a_2, a_3\}$, $va_2 = \{a_4, a_5\}$, $va_3 = \{a_6, a_7\}$). Thus, WfMS or KMS can disseminate process knowledge at the granularity suitable for marketers to serve their customers.

2.2. Basic definitions: base process and process-view

Fig. 2 illustrates how the components of our model are related. To differentiate the terminology used in base process and process-view, this work uses the terms virtual activity/dependency for the process-view while the terms base activity/dependency are used for the base process. A virtual activity is an abstraction of a set of base activities and corresponding base dependencies. A virtual dependency connects two virtual activities in a process-view.

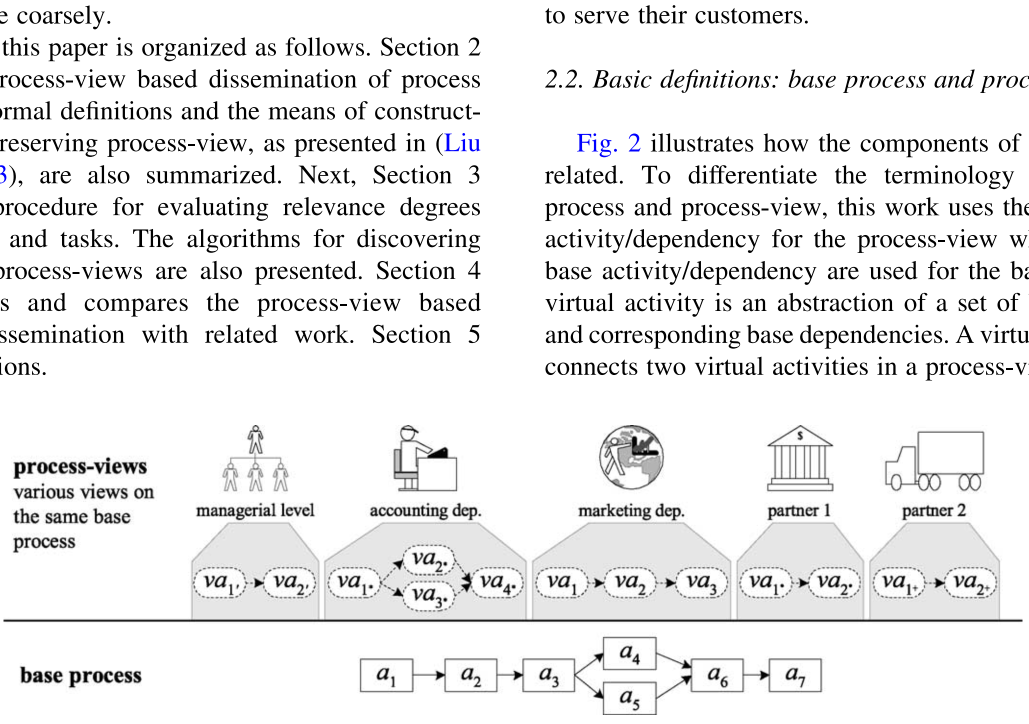


Fig. 1. Process-view based dissemination of process knowledge.

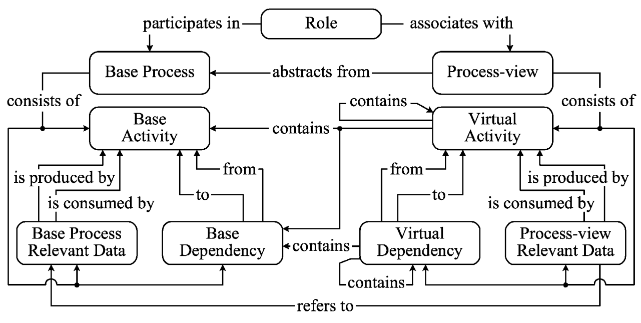


Fig. 2. Process-view model.

Fig. 3 shows an example of base process, where the split and join structures are defined by the Workflow management coalition (WfMC) (Workflow management coalition, 1998). *AND-split*: An activity splits into multiple parallel activities that are all executed. *XOR-split*: An activity splits into multiple mutually exclusive alternative activities, only one of which is followed. *AND-join*: Multiple parallel executing activities join into a single activity. *XOR-join*: Multiple mutually exclusive alternative activities join into a single activity. The following summarizes basic definitions of base process and process-view. Please refer the work of Liu and Shen (2003) for detailed definitions, semantics and examples.

Definition 1. (Base process) A base process *BP* is a 2-tuple $\langle BA, BD\rangle$, where

1. *BD* is a set of dependencies. A dependency is denoted by *dep* (x, y, C). Condition *C* represents the constraints, such as time or events, that determine whether routing can proceed from activity *x* to activity *y*.
2. *BA* is a set of activities. An activity is a 3-tuple $\langle$*SPLIT_flag*, *JOIN_flag*, *SC*$\rangle$, where (a) *SPLIT_flag*/*JOIN_flag* may be 'NULL', 'AND', or 'XOR'. NULL indicates that this activity has a single outgoing/incoming dependency (Sequence). Given multiple outgoing dependencies, AND indicates all succeeding branches are followed (AND-split), while XOR indicates only one succeeding branch is followed (XOR-split). Given multiple incoming dependencies, AND indicates that this activity can be started if all incoming dependencies have satisfied condition (AND-join), while XOR indicates that this activity can be started if one of the incoming dependencies has satisfied condition (XOR-join). (b) *SC* is the starting condition of this activity. A workflow engine evaluates *SC* to determine whether this activity can be started. If *JOIN_flag* is NULL, *SC* equals the condition associated with its incoming dependency. If *JOIN_flag* is AND/XOR, *SC* equals Boolean AND/XOR combination of the conditions of all incoming dependencies.
3. For $x, y \in BA$: (a) If there is a path from *x* to *y* in *BP*, then the ordering of *x* is higher than *y*, i.e. *x* precedes *y*. Their ordering relation in *BP* is denoted by $x > y$ or $y < x$. (b) If no path exists from *x* to *y* or from *y* to *x* in *BP*, then *x* and *y* are ordering independent, i.e. *x* and *y* proceed independently. Their ordering relation in *BP* is denoted by $x\infty y$.

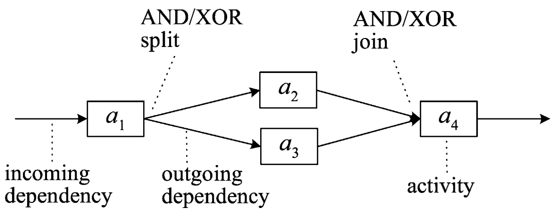


Fig. 3. Sample process.

Definition 2. (Process-view) A process-view is a 2-tuple $\langle VA, VD\rangle$, where (1) *VA* is a set of virtual activities. (2) *VD* is a set of virtual dependencies. (3) Analogous to base process, $\forall va_i, va_j \in VA$, the ordering relation between va_i and va_j may be '$>$', '$<$', or '∞'.

2.3. Order-preserving process-views

According to the different properties of a base process, various approaches can be developed to derive a process-view. A novel order-preserving approach to derive a process-view from a base process has been presented by Liu and Shen (2003) and is summarized below. Notably, this summary only presents the case that a base process does not contain loops. Please refer Liu and Shen (2003) for the case that a base process contains loops. The order-preserving approach ensures that the original execution order in a base process is preserved. A legal virtual activity in an order-preserving process-view must follow three rules: membership, atomicity, and order preservation rules (Liu & Shen, 2003). Simply, a virtual activity/dependency is an aggregation of a set of base activities/dependencies. The following defines virtual activities and virtual dependencies in an order-preserving process-view.

Definition 3. (Virtual activity) For a base process $BP = \langle BA, BD\rangle$, a virtual activity *va* is a 5-tuple $\langle A, D,$ *SPLIT_flag*, *JOIN_flag*, *SC*$\rangle$, where

1. *A* is a nonempty set, and its members follow three rules:
   a. Members of *A* are base activities that are also members of *BA* or other previously defined virtual activities that are derived from *BP*.
   b. *va* is started if one member activity is started, and is completed if all member activities are completed. (Note: this is a simplified expression of the original definition by Liu and Shen (2003))

   c. For any $x \in BA$, $x \notin A$, $\Re \in \{<, >, \infty\}$: if existing a $y \in A$ such that $x \Re y$ holds in BP, then $x \Re z$ holds in BP for all $z \in A$. That means the ordering relations between x and all members (base activities) of A are identical in BP.
2. $D = \{dep(x, y, C_{xy}) | \ dep\ (x, y, C_{xy}) \in BD$ and $x, y \in A\}$.
3. *SPLIT_flag* may be 'NULL' or 'MIX'. NULL suggests that *va* has a single outgoing virtual dependency while MIX indicates that *va* has multiple outgoing virtual dependencies.
4. *JOIN_ flag* may be 'NULL' or 'MIX'. NULL suggests that *va* has a single incoming virtual dependency while MIX indicates that *va* has multiple incoming virtual dependencies.
5. SC is the starting condition of va.

The *SPLIT_flag* and *JOIN_flag* cannot simply be described as AND or XOR since *va* is an abstraction of a set of base activities that may be associated with different ordering structures. Therefore, MIX is used to abstract the complicated ordering structures. A WfMS evaluates *SC* to determine whether *va* can be started. Members of *A* are called *va*'s member activities, and members of *D* are called *va*'s member dependencies. To save space, the abbreviated notation $va = \langle A, D \rangle$ is employed below to represent a virtual activity.

Definition 4. (Virtual dependency) For two virtual activities $va_i = \langle A_i, D_i \rangle$ and $va_j = \langle A_j, D_j \rangle$ that are derived from a base process $BP = \langle BA, BD \rangle$, a virtual dependency from va_i to va_j is $vdep\ (va_i, va_j, VC_{ij}) = \{dep\ (a_x, a_y, C_{xy}) | \ dep\ (a_x, a_y, C_{xy}) \in BD, a_x \in A_i, a_y \in A_j\}$, where the virtual condition VC_{ij} is a Boolean combination of C_{xy}.

3. Discovering role-relevant process-views

A role-based approach is proposed herein to discover process-views suitable for workflow participants. This section first describes the measurement of relevance degrees between roles and tasks, and then proposes three algorithms to generate automatically process-views.

3.1. Role-task relevance

The crucial part of process-view design is finding the relevance degree between each activity (task) and each role in a given base process. Based on these relevance degrees, process designers can define appropriate process-views for different organizational roles. Consequently, WfMS or KMS can disseminate process knowledge at a suitable granularity to participants, i.e. more relevant parts are presented with a finer resolution and less relevant parts are coarser.

Process designers can directly evaluate the relevance degrees between tasks and roles. However, inconsistency and bias may occur without criteria for evaluating relevance degrees. Moreover, as the number of tasks grows, the evaluation becomes excessively complex. Besides, knowledge embedded in access control systems is not utilized. WfMC has defined several workflow operations for controlling and monitoring task execution (Workflow Management Coalition, 1995). For example, *WMFetchActivityInstanceState* returns an activity state, and *WMReassignWorkItem* reassigns an activity from one participant to another. Moreover, some workflow applications may provide extended operations such as *ProgressTracking* and *PerformerSelection*. Whether a role is authorized to perform an operation on a task object is determined by the organization's security systems, or more specifically, by role-based access control systems (Ferraiolo et al., 2001). Hence, this work applies these different operations as the criteria for evaluating degrees of relevance since their number is finite, and the relationships among tasks, roles and operations are specified by security constraints.

3.1.1. Relevance evaluation based on workflow operations

The associations between roles and tasks are established based on workflow operations. Each task is associated with a set of operations that participating roles may perform on it. For example, a role may perform *ProgressTracking* operation on a task. Therefore, if participating roles evaluate the relevance degrees of these operations, then the role-task associations can be quantified. However, security constraints are such that roles cannot perform operations on tasks arbitrarily. Accordingly, the relevance degrees between roles and tasks are evaluated based on the usage of workflow operations and the access control policies.

Fig. 4 depicts the framework for evaluating role-task relevance. For example, if the relevance degree of role *r* with respect to operation *op* is 0.4 (role profile); task *t* supports operation *op* (task profile), and *r* is authorized to perform *op* on *t* (permission rule), then the relevance degree between role *r* and task *t* is assessed as 0.4. The main components are as follows.

Task profile. Different tasks may be associated with different operations. A task profile records which operations can be performed on a given task.

Role profile. A role has different degrees of relevance to its authorized operations. A role *r*'s profile includes a set of pairs ⟨operation *op*, relevance degree *deg*⟩, which indicates that the relevance degree of role *r* with respect to operation *op* is *deg*.

Permission rules. These rules are derived from access control systems, and govern which role is authorized to perform which workflow operation on which task. Namely, a permission rule ⟨role *r*, task *t*, operation *op*⟩ states that role *r* is authorized to perform operation *op* on task *t*.

Task execution log. Each record in this database is a 3-tuple ⟨TaskInstNo, RoleInstNo, operation name⟩, where

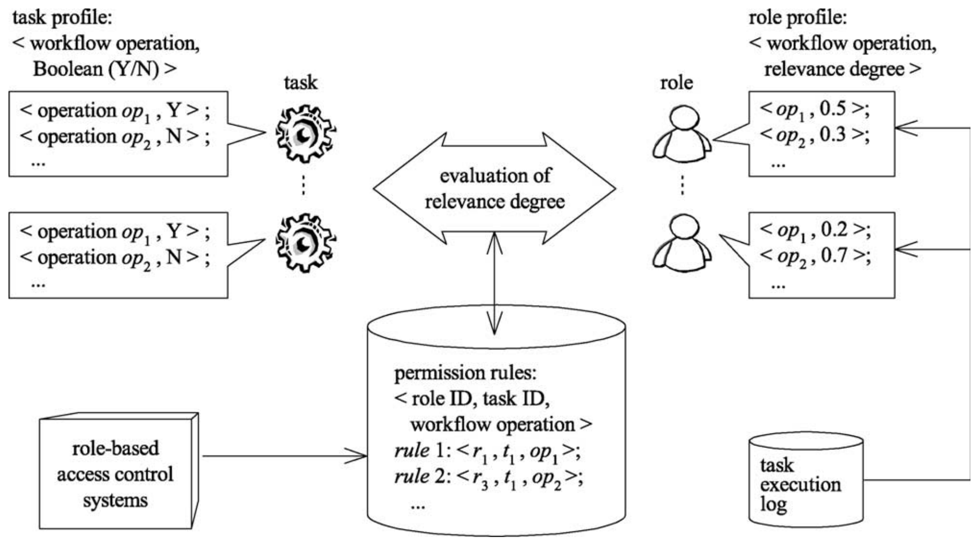

task profile:
< workflow operation, Boolean (Y/N) >
task
< operation op_1 , Y > ;
< operation op_2 , N > ;
...
evaluation of relevance degree
role
role profile:
< workflow operation, relevance degree >
< op_1 , 0.5 >;
< op_2 , 0.3 >;
< op_1 , 0.2 >;
< op_2 , 0.7 >;
permission rules:
< role ID, task ID, workflow operation >
rule 1: < r_1 , t_1 , op_1 >;
rule 2: < r_3 , t_1 , op_2 >;
role-based access control systems
task execution log

Fig. 4. Evaluating role-task relevance.

TaskInstNo/RoleInstNo is a task/role instance identifier. A log record states that a role instance has performed a specific operation on a task instance.

Construction of role profile. Initially, role profiles are constructed by analyzing these logs. Let M denote the number of tasks on which operation op is performed, and let N denote the number of tasks on which role r performs operation op. The default degree of relevance of r with respect to op is N/M. For example, if the *ProgressTracking* operation is performed on 100 tasks, and the *IT manager* executes 20 s of them, then, the relevance degree of *IT manager* with respect to the *ProgressTracking* operation is 0.2. Several factors can also be considered to enrich the role profiles. Intuitively, a higher frequency or cost associated with an operation performed by a role corresponds to a higher relevance degree of the operation to the role. Let Q denote the number of tasks executed by role r. The frequency that r performs operation op is N/Q. Cost can be measured by the time and money consumed by roles in performing operations on tasks. In summary, different statistics can be calculated from historical data. Multi-criteria decision making methods can be adopted to derive the best combination of these statistics. Moreover, relevance feedback from workers further adjusts the relevance values to fit their information needs.

Generally, the degree of relevance is more easily described qualitatively in linguistic terms than by numerical quantities. The notion of *linguistic variables* proposed by Zadeh (1965) may be useful in solving such a problem. A linguistic variable is a variable whose values are words or sentences in a natural language. For example, the values of the linguistic variable 'relevance' can be described using five linguistic scales: irrelevant, barely relevant, moderately relevant, very relevant and extremely relevant. Fuzzy numbers, i.e. fuzzy sets (Zadeh, 1975) defined on the real line $\Re$, can represent these linguistic scales. Linguistic variables and fuzzy numbers allow process designers to use linguistic measures to assess degrees of relevance, and overcome the vagueness of human decision-making.

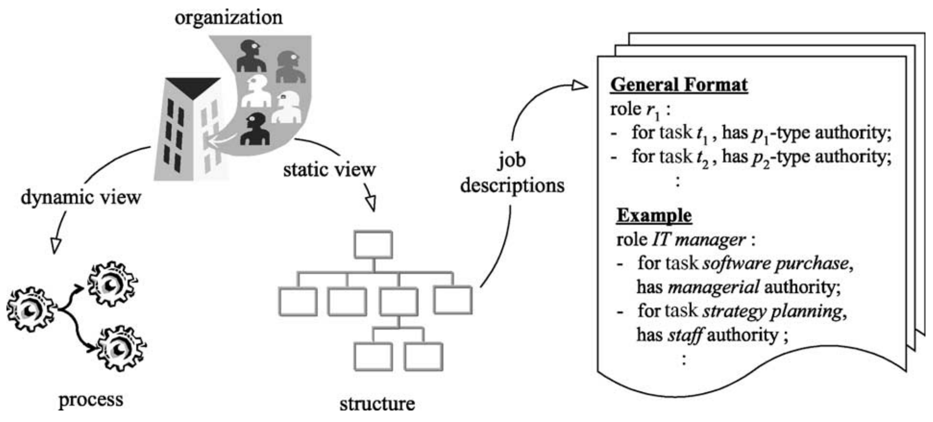
However, the relevance degree is represented by crisp values herein to keep the focus on the evaluation procedure. The interested reader may refer Shen, Tzeng, and Liu (2003), wherein fuzzy linguistic quantifiers are used to evaluate the appropriateness of workers for performing tasks in workflow systems.

Evaluation of role-task relevance. The relevance degrees between a given role and tasks of a workflow can be derived from the role/task profiles and permission rules. For example, given a permission rule ⟨role r, task t, operation op_1⟩, role r's profile {⟨ op_1, 0.4⟩, ⟨ op_2, 0.8⟩, …}, and task t's profile {⟨ op_1, Y⟩, ⟨ op_1, Y⟩,…}, the relevance degree between role r and task t is 0.4. However, if another rule ⟨ r, t, op_2⟩ is added, then the relevance degree is $\max(0.4,0.8) = 0.8$. That is, the maximum relevance degree is selected when a role is authorized to perform multiple operations on a single task.

Finally, process designers use the granular threshold (described in Section 3.2.1) to determine the granularity of generated process-views. The process-view definition tool automatically derives process-views according to role-task relevance and threshold value. Section 3.1.2 presents detailed algorithms.

3.1.2. Relevance evaluation based on organizational authority

Different types of organizational authority are alternative evaluation criteria. An organization can be viewed statically and dynamically, as shown in Fig. 5. The design of an organizational structure legitimately distributes authority and responsibility to task holders. The resultant job descriptions specify the relationships among roles, tasks, and authority. Fig. 5 also shows the general format and examples of job descriptions. Therefore, using organizational authority as criteria for evaluating the role-task



General Format
role r_1 :
- for task t_1, has p_1-type authority;
- for task t_2, has p_2-type authority;
:

Example
role *IT manager* :
- for task *software purchase*, has *managerial* authority;
- for task *strategy planning*, has *staff* authority ;
:

Fig. 5. Different views of an organization and job descriptions.

relevance may be friendly and meaningful for organizational managers and process designers.

Basically, authority is a higher level concept than workflow operations. That is, ‘a role has authority over a task’ means that ‘the role is authorized to perform a set of operations on the task’. Each type of authority corresponds to a set of related workflow operations. Therefore, given a mapping table that states the correspondence between different types of authority and workflow operations, process designers can evaluate role-task relevance based on the exertion of authority. In the previous framework depicted in Fig. 4, role-task associations are established based on the performance of operations. With the mapping table, operation-based associations are transformed straightforwardly to authority-based operations.

3.2. Algorithms for generating process-views

Algorithms are proposed to generate automatically process-views based on the relevance degrees derived as described above. This section first presents three algorithms to generate all legal virtual activities (tasks). Then, the generation of virtual dependencies is discussed.

3.2.1. Generating virtual activities of a process-view

Algorithm 1 determines the virtual activity set of a process-view definition from a base process. The objective of this algorithm is to discover virtual activities whose relevance degree approximates the granular threshold TH, as specified by the process designer. The process of generating virtual activities begins with the *highest ordering* activities in the base process. When the total relevance degree of a set of base activities approximates the granular threshold TH, a virtual activity is found. The above steps are repeated against residual base activities until virtual activities cover all base activities of the base process. Consequently, the virtual activity set of the target process-view is obtained. The following explains four important parts of algorithm 1, including the granular threshold, the total relevance degree, the virtual activity generator (algorithm 2), and the verification of order-preserving property (algorithm 3).

Algorithm 1. (The generation of virtual activity set)

```
1:  Input: a base process $BP = \langle BA, BD \rangle$.
2:  Output: the virtual activity set ($VA$) of a process-view $VP = \langle VA, VD \rangle$
3:  Require: $f_{RD}(a_i) \le$ TH for all activities of $BP$.
4:
5:  procedure main(base process $BP = \langle BA, BD \rangle$)
6:  begin
7:     $i \leftarrow 0$;
8:     repeat
9:        $i \leftarrow i + 1$;
10:       $va_i = \langle A_i, D_i \rangle \leftarrow \langle \varnothing, \varnothing \rangle$;
11:       residual activity set $RAS \leftarrow BA - \{x \mid x$ belongs to one of $A_j, j = 1..i\}$;
12:       select a highest ordering activity $x$ from $RAS$;
13:       $va_i \leftarrow$ VAGenerator($x$, $RAS$, $BP$);
14:    until for all $x \in BA$, $x$ belongs to one of $A_j, j = 1..i$
15:    return $VA \leftarrow \{ va_j, j = 1..i \}$;
16: end
```

Granular threshold TH. This parameter determines the granularity of generated process-view. A virtual activity is an aggregation of a set of base activities. When the sum of the relevance degrees of some base activities approximates the threshold value, these activities can form a virtual activity, which is seen as relevant enough to the role. Process designers subjectively specify TH according to their personal experience, expertise, and the application domain. A larger TH corresponds to the generation of fewer virtual activities (and more base activities included in a virtual activity). Notably, TH must be larger than or equal to the maximum relevance degree between roles and tasks (line 3).

Total relevance degree F_{RD}( ). Simply, if the function f_{RD} (base activity *ba*) returns the relevance degree of *ba* and function F_{RD} (activity set *A*) returns the total relevance degree of *A*, then $F_{RD}(A) = \sum f_{RD}(a_i)$ for all $a_i \in A$.

However, ordering structures should be considered. Consider an example in which the threshold value equals 1, a base activity a_1 splits into a_2 and a_3, and their relevance degrees to a role are 0.6, 0.3, and 0.4, respectively. Clearly, the three activities do not belong to the same virtual activity, according to the above formula ($f_{RD}(a_1) + f_{RD}(a_2) + f_{RD}(a_3) >$ threshold). However, during run-time, all activities that follow *AND-split* are executed while only one activity is executed after *XOR-split*. Thus, if the split structure is *XOR-split* in the example, then the maximum relevance degree of the three activities is expected to be $0.6 + 0.4$, rather than $0.6 + 0.3 + 0.4$ or $0.6 + 0.3$. Namely, the total relevance degree $F_{RD}(A)$ is the maximum expected value of $\sum f_{RD}(a_i)$ for all $a_i \in A$. The above discussion also holds for *XOR-join*.

Algorithm 2. (The generation of a virtual activity whose total relevance degree approximate granular threshold)

```
1:  procedure VAGenerator (seed activity $a$, residual activity set $RAS$, $BP = \langle BA, BD \rangle$)
2:  begin
3:     $va = \langle A, D \rangle \leftarrow \langle \{a\}, \emptyset \rangle$;
4:     repeat
5:       temp activity set $TAS \leftarrow A$;
6:       adjacent activity set $AAS \leftarrow \{x \mid x, y \in RAS, x \notin A, y \in A$, and $dep(x, y) \in BD\}$;
7:       while $AAS$ is not empty do
8:           select an activity $x$ from $AAS$;
9:           remove $x$ from $AAS$;
10:          $A_{tmp} \leftarrow OP(A \cup \{x\}, BP)$;            /* check order-preserving property */
11:          if ($F_{RD}(A_{tmp}) \leq TH$) and ($A_{tmp} \subseteq RAS$) then    ,    /* check threshold */
12:              $A \leftarrow A_{tmp}$;
13:              $AAS \leftarrow AAS - \{y \mid y \in AAS \cap A\}$;
14:          end if
15:      end while
16:    until $A = TAS$
17:    a dependency set $D \leftarrow \{dep(x, y) \mid x, y \in A$, and $dep(x, y) \in BD\}$;
18:    return $va = \langle A, D \rangle$;
19: end
```

Virtual activity generation. Algorithm 2 discovers a virtual activity that contains a given base activity and has approximately the maximum degree of relevance below the granular threshold TH. For a virtual activity $va = \langle A, D \rangle$, the members of A must be identified first, since D can be derived from A. Initially, A contains only the given activity a (line 3). Algorithm 2 then determines whether the activities adjacent to members of A can be added to maximize its total relevant degree ($F_{RD}(A)$) to approximate TH. A is updated during the *while* loop (lines 7 ~ 15) by adding the adjacent activities that cause A to satisfy three conditions: A conforms to the order-preserving property (line 10, see Algorithm 3); total relevance degree of A does not exceed the threshold ($F_{RD}(A_{tmp}) \leq TH$); A does not overlap with previously derived virtual activities ($A_{tmp} \subseteq RAS$). The *repeat-until* loop (lines 4 ~ 16) continues until no other adjacent activity is added to A (line 16, $A = TAS$), i.e. no more adjacent activity can be added to A while still satisfying the threshold limit and maintaining order-preservation.

Following the determination of A, the members of D are those dependencies whose succeeding and preceding activities are both members of A (Definition 3.1). Thus, the virtual activity of $va = \langle A, D \rangle$ is generated (lines 17 ~ 18).

Algorithm 3. (The generation of a legal virtual activity)

```
1:  procedure OP (activity set $AS$, base process $BP = \langle BA, BD \rangle$)
2:  begin
3:     repeat
4:       temp activity set $TAS \leftarrow AS$;
5:       adjacent activity set $AAS \leftarrow \{x \mid x, y \in BA, x \notin AS, y \in AS$ and $dep(x, y) \in BD\}$;
6:       while $AAS$ is not empty do
7:           select an activity $x$ from $AAS$;
8:           remove $x$ from $AAS$;
9:           if $\exists y, z \in AS$, and $x, y, z \in BA$, $\Re \in \{<, >, \infty\}$,
10:              such that ( $x \Re y$ holds in $BP$ but $x \Re z$ does not holds in $BP$ ) then
11:          /* the ordering relations between $x$ and all base activities of $AS$ are
12:              not identical */
13:              add $x$ to $AS$;
14:          end if
15:      end while
16:    until $AS = TAS$
17:    return $AS$;
18: end
```

Verification of order-preserving property. For a given activity set AS, algorithm 3 is capable of obtaining the member activities of a minimum and order-preserving virtual activity. According to the definition of a virtual activity (Definition 3.1c), a legal virtual activity $va = \langle A, D \rangle$ must satisfy the order-preserving condition: the ordering relations between x and all members of A that belong to BP are identical for any base activity $x \in BA$ and $x \notin A$.

An activity x, $x \notin AS$, is included in A for order preservation. AS is obviously a starting point for identifying x. The algorithm begins from AS, by initializing an activity set TAS that equals AS, to check whether AS is a legal (i.e. order-preserving) virtual activity. If AS is not legal, AS is updated by including activities that violate the order-preserving condition. To determine which of the activities should be added into AS to form a legal and minimal virtual activity, the algorithm considers the activities that are adjacent to members of AS. The algorithm determines whether adjacent activities of AS satisfy the order-preserving condition (lines 9 ~ 12). AS is updated during the *while* loop (lines 6 ~ 15), by adding adjacent activities that violate the order-preserving condition. If AS is updated, the *repeat-until* loop is repeated to check the order-preserving condition. The *repeat-until* loop (lines 3 ~ 16) continues to repeat until no more adjacent activity is added into AS (line 16, $AS = TAS$), i.e. all adjacent activities of AS satisfy the order-preserving condition. Finally, AS is a legal virtual activity.

Since this virtual activity conforms to Definition 3, it is a legal virtual activity. Moreover, the algorithm checks the ordering relations from adjacent activities, creating a minimal virtual activity.

Naming virtual activities. Since virtual activities are aggregations of base activities, the aggregation must be given a meaning. Relevance degrees can be used to interpret the generated virtual activities. A higher degree of a base activity corresponds to its stronger influence on the formation of the virtual activity. Therefore, the relevance degrees can be used to determine which member activities influence the formation of a virtual activity, and a name or label can be assigned to that virtual activity. However, no absolute guidelines determine how high a relevance degree must be before a given member activity is called influential on the formation of a virtual activity. Usually, only member activities with the top N relevance degrees are considered in the naming of a virtual activity.

3.2.2. Generating virtual dependencies

Following the generation of all virtual activities, the virtual dependencies are derived based on Definition 4. Given the virtual activity set VA of a process-view VP derived from a base process BP, whether or not a virtual dependency is associated with two virtual activities can be determined. For any two distinct virtual activities $va_i = \langle A_i, D_i \rangle$ and $va_j = \langle A_j, D_j \rangle$: if $\exists a_x \in A_i$ and $\exists a_y \in A_j$ such that $dep(a_x, a_y, C_{xy})$ exists in BP, then $vdep(va_i, va_j, VC_{ij})$ exists in VP and $dep(a_x, a_y, C_{xy})$ is a member of $vdep(va_i, va_j, VC_{ij})$. After checking each base dependency, all virtual dependencies and their members can be derived.

The derivation of VC of virtual dependencies and *SPLIT/JOIN_flag* of virtual activities are omitted herein and can be found in the work of Liu and Shen (2003).

3.3. Prototype systems

This section briefs the architecture of our prototype system as presented by Liu and Shen (2003). The prototype is implemented mainly using the Java 2 platform, Enterprise Edition (J2EE) (Kassem, 2000). J2EE offers rich and uniform application programming interfaces (API) to access diverse business systems. For example, the Java Database Connectivity (JDBC) API gives a vendor-independent interface to different DBMSs. Therefore, the prototype is independent of any specific implementation of a DBMS, directory service, or messaging service. The prototype is built on BEA WebLogic (our J2EE server) (Gomez & Zadrozny, 2000), Microsoft SQL Server (our DBMS) (Vieira, 2000), and Microsoft Active Directory (MSAD, our directory service) (Lowe-Norris, 2000).

Fig. 6 shows the system's architecture. The prototype system uses the Java Naming and Directory Interface (JNDI) API to locate participants that registered in MSAD. The role designer maps role profiles onto the organizational units, groups, and users defined in MSAD. Role profiles are

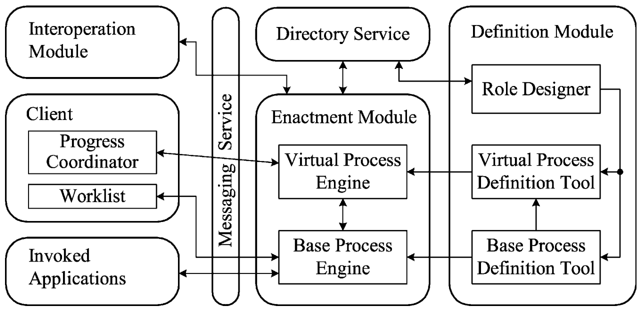


Fig. 6. Architecture of the prototype system.

consulted during base/virtual process design. A process modeler employs the role designer and the virtual/base process definition tool to specify workflow participants (internal workers and trading partners), base processes and process-views. The role-based procedure for deriving process-views as elucidated in previous sections is implemented in the virtual process definition tool.

During run-time, the enactment module consults MSAD to obtain clients' responsibilities, and then submits/receives messages to clients' work-lists or progress coordinators. In this prototype, the enactment module communicates with the clients, the interoperation module, and the invoked applications through the Java Message Service (JMS) API. JMS supports the publish/subscribe asynchronous communication mechanism in the process-view model. The client-side work-list presents the activities that must be performed by the client. The progress coordinator displays abstracted progress information of the workflow in which the client participates. Role-relevant process knowledge is disseminated to workers through client-side progress coordinators.

4. Related work

4.1. Dissemination of working knowledge

Just-in-time knowledge delivery aims to provide workers with the knowledge required to perform a task when they need it (Cole, Fischer, & Saltzman, 1997). That is, working information is delivered automatically to workers according to their working context (situation) when they are actually performing a task. Context-aware information delivery involves similar ideas, e.g. (Brown & Jones, 2001; Fischer & Ye, 2001). Broadly, these studies characterize information sources and working context by common features, and then deliver relevant information to workers based on the feature matching between context and information sources. In some information retrieval based approaches, each working context is associated with a query predicate which is a Boolean combination of features used to describe information sources. When workers enter a specific context, a relevant query is executed and resultant information is recommended to them. Generally, these approaches must establish comprehensive metadata.

Using similar ideas, several approaches have been developed to deliver relevant and necessary documents to workers to enable them to accomplish their tasks in a workflow environment (Abecker et al., 2000; Reimer, Margelisch, & Staudt, 2000; Staaba & Schnurr, 2000). However, such approaches suggest task-specific information, rather than process-oriented knowledge. Being informed about a whole process is important since workers are responsible for the outcomes of their participating workflow (Hammer & Champy, 1993). Therefore, this work focuses on providing workflow-scope information. With the support of process knowledge, workers can, for example, identify process bottlenecks and monitor progress status. Both workflow and task-scope knowledge are necessary for effective KM.

4.2. Process mining

Workflow mining aims to discover process definitions from historical sequences of executed tasks, as described for example in Refs. (Agrawal, Gunopulos, & Leymann, 1998; Datta, 1998; Hwang & Yang, 2002; Weijters & Van der Aalst, 2001). These studies focused on evaluating the correlation among tasks to infer task orderings. Meanwhile, Chun et al. proposed a domain knowledge-based approach to automating the generation of workflow definitions (Chun, Atluri, & Adam, 2002). They assumed that given domain ontologies have defined all possible services and corresponding implementations (tasks and their orderings). Thus, a workflow definition is automatically generated based on user-selected services. In contrast to the above references, this work discovers appropriate process abstractions from a given base process definition for workflow participants. Therefore, this work focuses on measuring the relevance between organizational roles and tasks. The definitions of role-relevant process-views are generated from a given base process, according to the resulting relevance degrees.

4.3. Procedure of process-view design

Our previous work (Liu & Shen, 2003) focused on process-view and order-preservation. In that work, process-views were designed through an interactive procedure. Process designers select some activities as a seed, and then the process-view definition tool discovers a legal virtual activity from the seed. This step is repeated until all virtual activities are identified. Notably, the selection of seeds is based on a designer's subjective assessment of the relevance between roles and tasks. This work abandons the above approach and proposes a new framework for evaluating role-task relevance to simplify process-view design. Although only operation usage is considered as an evaluation criterion, other attributes can be easily added to the role/task profile to enable advanced evaluation of relevance. This work also proposes new algorithms to generate automatically process-views based on the results of evaluation.

The aim of the proposed algorithm is to discover virtual activities whose relevance degree approximates the granular threshold. The algorithm starts from the first activities (start nodes) of a process and merges base activities into virtual activities. However, several possible approaches to finding process-views remain. For example, base activities whose relevance degree exceeds a threshold constitute individual virtual activities. Then, other base activities are merged into those virtual activities. For another example, the number of required virtual activities is given and then base activities are merged. These approaches are worthy of further exploration.

5. Conclusions and future work

This work presents a process-view based dissemination of process knowledge. Task granularity of delivered information is adapted to the needs of workflow participants. Workers can thus obtain helpful views of a large and complex workflow. This work proposes a systematic procedure for measuring the degrees of relevance between roles and tasks to support the discovery of role-relevant process-views. Role-based access control systems formally authorize roles to perform workflow operations on tasks. The relationships among roles, tasks and operations are listed in permission rules. Therefore, this work uses workflow operations as criteria to evaluate role-task relevance. Role-relevant process-views are automatically generated based on evaluation results using the proposed algorithms. Accordingly, KMS or WfMS can supply workers with process-oriented knowledge at the granularity suitable for their organizational role.

Future work will address three issues. First, a real case should be considered to validate the proposed approach. Second, only role characteristics (role profile) are considered herein. However, workers may have various features, such as working expertise and experience, so future work should construct user profiles to assist WfMS or KMS in providing more personalized knowledge dissemination. Finally, the process-view model was applied to coordinate inter-organizational workflows (Liu & Shen, 2003). The approach proposed herein may be extended to discover business-to-business process-views by analyzing contracts of cooperation.

Acknowledgements

This research was supported by National Science Council of the Republic of China under the grant NSC 92-2416-H-009-010.

References

Abecker, A., Bernardi, A., Maus, H., Sintek, M., & Wenzel, C. (2000). Information supply for business processes: Coupling workflow with document analysis and information retrieval. *Knowledge-Based Systems*, *13*(5), 271–284.

Agrawal, R., Gunopulos, D., & Leymann, F. (1998). Mining process models from workflow logs. *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, March23–27*.

Brown, P. J., & Jones, G. J. F. (2001). Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. *Personal and Ubiquitous Computing*, *5*(4), 253–263.

Chun, S. A., Atluri, V., & Adam, N. R. (2002). Domain knowledge-based automatic workflow generation. *Proceedings of 13th International Conference on Database and Expert Systems Applications (DEXA'02) Aix-en-Provence, France, September 2–6*, 81–93.

Cole, K., Fischer, O., & Saltzman, P. (1997). Just-in-Time knowledge delivery. *Communications of the ACM*, *40*(7).

Datta, A. (1998). Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. *Information Systems Research*, *9*(3), 275–301.

Davenport, T. H., & Prusak, L. (1998). *Working knowledge: How organizations manage what they know* (1st ed). Boston: Harvard Business School Press.

Edvinsson, L., & Malone, M. (1997). *Intellectual capital: Realizing your company's true value by finding its hidden brainpower*. NY, USA: Harper Business Publishers.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and Systems Security*, *4*(3), 224–274.

Fischer, G., & Ye, Y. (2001). Exploiting context to make delivered information relevant to tasks and users. *Proceedings of Workshop on User Modeling for Context-Aware Applications at the 8th International Conference on User Modelling, Sonthofen, Germany, July 13–16*.

Georgakopoulos, D., Hornick, M., & Sheth, A. (1995). An overview of workflow management-from process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, *3*(2), 119–153.

Gomez, P., & Zadrozny, P. (2000). *Professional J2EE programming with be a web logic server*. Birmingham: Wrox Press.

Hammer, M., & Champy, J. (1993). *Reengineering the corporation*. NY, USA: Harper Business Publishers.

Hwang, S.-Y., & Yang, W.-S. (2002). On the discovery of process models from their instances. *Decision Support Systems*, *34*(1), 41–57.

Kassem, N. (2000). *Designing enterprise applications with the Java 2 platform*. Boston: Addison-Wesley, Enterprise Edition.

Leymann, F., & Altenhuber, W. (1994). Managing business processes as an information resource. *IBM Systems Journal*, *33*(2), 326–348.

Liao, S.-h. (2002). Knowledge management technologies and applications-literature review from 1995 to 2002. *Expert Systems with Applications*, *25*(2), 155–164.

Liu, D.-R., & Shen, M. (2003). Business-to-business workflow interoperation based on process-views. *Decision Support Systems*, (in press).

Liu, D.-R., & Shen, M. (2003). Workflow modeling for virtual processes: An order-preserving process-view approach. *Information Systems*, *28*(6), 505–532.

Lowe-Norris, A. G. (2000). *Windows 2000 Active Directory*. Cambridge, USA: O'Reilly.

Reimer, U., Margelisch, A., & Staudt, M. (2000). EULE: A knowledge-based system to support business processes. *Knowledge-Based Systems*, *13*(5), 261–269.

Shen, M., Tzeng, G.-H., & Liu, D.-R. (2003). Multi-criteria task assignment in workflow management systems. *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36'03), Big Island, Hawaii, USA, January 6–9*, 6–9. pp. 202–210.

Staaba, S., & Schnurr, H.-P. (2000). Smart task support through proactive access to organizational memory. *Knowledge-Based Systems*, *13*(5), 251–260.

Vieira, R. (2000). *Professional SQL server 2000 programming*. Birmingham, UK: Wrox Press.

Weijters, A. J. M. M., & Van der Aalst, W. M. P. (2001). Process mining: Discovering workflow models from event-based data. *Proceedings of the 13th Belgium–Netherlands Conference on Artificial Intelligence (BNAIC 2001), Amsterdam, Netherlands, October*, 25–26.

Workflow Management Coalition, *The Workflow Reference Model*, Technical report WfMC TC-1003, Jan. 19, 1995

Workflow Management Coalition, *Workflow Management Application Programming Interface (Interface 2 and 3) Specification*, Technical report WFMC-TC-1009, 1998

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning. Parts 1, 2, and 3. *Information Sciences*, *8*(3), 199–249. see also: 8(4), 301–357, 1975, 9(1), 43–80, 1976.