Dynamic Routing and Operational Controls in Workflow Management Systems

Akhil Kumar *

College of Business, Campus Box 419
University of Colorado
Boulder CO 80309-0419

J. Leon Zhao

School of Business and Management
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Businesses around the world are paying more attention to process management and process automation to improve organizational efficiency and effectiveness. In this paper, we describe a general framework for implementing dynamic routing and operational control mechanisms in Workflow Management Systems (WMSs). The framework consists of three techniques: workflow control tables, sequence constraints, and event-based workflow management rules. Our approach offers several unique features that are missing in commercial workflow management systems: (1) it provides more flexibility in process modeling and control; (2) it permits rework on an ad hoc basis; (3) it handles exceptions to routing and operational controls; and (4) it exploits parallelism to increase system throughput and response time. Finally, the workflow management techniques are applied to the case of consumer loan management and compared with other approaches based on static routing.

1 Introduction

The last few years have seen widespread application of business process reengineering in the corporate world [10, 18, 24]. As a result, business organizations have become more and more dependent upon highly automated business processes. Management of business processes in an organizational setting is now commonly referred to as *workflow management*. Information systems that support workflow management are called Workflow Management Systems (WMS). At the moment, more than 250 workflow management systems are under development [41]. Well known example systems are Lotus Notes [42], ActionWorkflow [28], and FlowMark [25, 26].

Presently, there is a surge of research activity in workflow models and languages [38], adding to the considerable interest in the development of workflow applications and systems in the recent past [5, 7, 16,

*The first author's work supported in part by the Research Committee, College of Business, University of Colorado. The second author would like to acknowledge the partial support by the Hong Kong Government Research Grant Council under contract HKUST6222/97H.

17, 25]. Furthermore, database systems are being extended to support workflow management [4, 12, 35], and conventional transactional models are being modified to encompass the complex coordination requirements of workflow applications [21, 34].

This new trend in the research community signifies that workflow management will have a tremendous impact on the next generation of information systems. However, workflow management as a field is still in its infancy, and there is an absence of conceptual frameworks and theoretical models for workflow management.

Workflow management involves the coordination and control of processes and activities of people and systems in an organization. According to [28], there are three kinds of activities in an organization: (1) *material processes* which transform physical components into products; (2) *information processes* where the flow of information occurs using sophisticated information technologies such as data flow analysis, database storage and retrieval, transaction processing, and network communication; and (3) *business processes* through which customers and suppliers interact with the organization to accomplish certain business goals. Activities in these three domains are indistinguishable at times, but have clear differences in concepts and technologies. Workflow management involves mainly the latter two processes.

Document processing has a fundamental place in workflow management as business documents are the common media of information processes and business processes. Essentially, workflow management involves the control of performance of operations on documents in terms of: (a) who can access which document; (b) what operations can be performed on a document by a worker; and (c) how the sequence of operations should be carried out by the various workers.

In a WMS, documents (e.g., purchase order request, travel authorization request, etc.) pass through various individuals. Each individual performs operations on the document such as: fill in some fields, modify fields, etc. For example, in a typical organization, an employee planning to travel on business would fill in a travel request authorization form, and provide information on the destination, dates of travel, and a tentative budget. This form then goes to a secretary who reviews it, and enters information on previous business travel by the employee. Next, the form is sent to the manager who approves or disapproves the requested amount. Alternatively, the manager may approve a smaller amount than requested. Finally, the form is returned to the individual concerned and the secretary to notify them of the decision.

In this simple application, there are several issues of control that must be considered. Since tasks are performed in a sequence, once the manager has approved a travel request, nobody else should be allowed to change the approved amount (or any other information). However, before the approval is received it is possible for the employee to modify the requested amount. Therefore, in general, an employee must not be given unconditional permission to make updates; rather it has to be controlled appropriately. Such cases are numerous.

Furthermore, there is a need for handling exceptions in WMS. For instance, there are occasions when a special measure is needed to speed up the processing of a particular document. An example is that a manager may decide that it is in the best interests of the company to bypass the normal sequence of workflow in order

to expedite the approval of a loan application. However, inadequate support has been reported in workflow applications for exceptions, resulting in poor quality and inflexible workflow management systems [15, 40].

Most off-the-shelf WMSs such as Lotus Notes [42] are very primitive in terms of authorization and operational control. There is no facility for sophisticated authorization and control beyond password checking for access to file folders and for read and write permissions. Large-scale WMSs such as FlowMark [25, 26] and ActionWorkflow [28] provide the basic program modules for network communication and database access, but extensive programming is needed to implement a system. Although there is no limit as to what system functions can be included in the programs, no theoretical models on workflow authorization and control were reported in the literature.

Traditional database systems cannot be used to address the authorization and control issues in workflow management. While they allow one to grant and revoke permissions in terms of who can read or update a table or set of fields [11], this is inadequate in a workflow system where the sequence in which operations are performed is important. For instance, there is no way to specify a constraint such as: *employee e1 can update a document D1 only if it has not been updated by anybody senior to him or her in the organization.* Here the permission to update for the employee is qualified in a temporal sense. Moreover, such constraints (or rules) could potentially be different for different documents.

Our research objective is to develop a framework for managing workflows in an efficient and orderly manner while allowing maximal degree of flexibility. Consequently, in this paper, we focus on supporting flexible control mechanisms using sequence constraints and workflow management rules. Previous workflow models are mostly based on static routing schemes (or routing maps) defined using graphical languages like ICN [13, 14] and Petri Nets [31, 44]. However, as we will show, these graphical process languages do not support naturally flexible routing and exception handling.

Although work in scheduling, planning and project management addresses routing problems in terms of job sequencing [6, 9, 20, 32, 33], the workflow sequence problem is different:

(1) Workflow management consists of tasks involving human operators who cannot be programmed like robots.

(2) Due to the inexact nature of manual operations, workflow management is difficult to formulate into an optimization problem.

(3) In business processes, changes occur very frequently, exceptions abound, and many versions of procedures co-exist. Therefore, new concepts and techniques are needed for workflow modeling and management.

The organization of this paper is as follows. The next section discusses preliminaries and provides an overview of our framework. In particular, it consists of three techniques. The first technique, detailed in Section 3, is based on workflow control tables and defines the authorized operations for each role in the workflow management system with respect to a document or a field. Section 4 describes the second technique which consists of means for imposing and enforcing constraints on the routing sequence of the workflow, i.e., the sequence in which a document is sent to various employees. In Section 5, the third technique called

event-based workflow management rules is described as a means for specifying unusual routing needs and other special actions. Section 6 gives a short case study of a consumer loan management application at a bank to illustrate the framework, while Section 7 provides a discussion and comparison with related research. Finally, Section 8 concludes the paper with a brief summary.

2 A Framework for Workflow Management Systems

In this section, we first define the basic terminology used in the paper.

- *Document* A statement or form (in electronic medium or hard copy) that is needed for a business transaction, e.g., a billing statement or a travel authorization form.
- *Field* Any data element contained in a document that has a numerical or symbolic value.
- *Worker* Any person who performs operations on documents using the WMS.
- *Role* A generic identifier for a group of workers, any one of whom may perform a task assigned to the "role". For example, the role may be order entry clerk, and the workers qualified to perform that role are Dave, Mike and Sue. Note that, in general, a given worker may be qualified to perform one or more roles. The WMS maintains a mapping from roles to workers.
- *Work basket* A unique, logical box (or in-tray) associated with a role. *Work basket* is the term used to denote a stack (or queue) of documents waiting to be processed by a worker in the corresponding role.
- *Operation* The smallest unit of work, e.g., entering a data value into the amount field on a form.
- *Task* A collection of operations that corresponds to a step in a common business process, e.g., filling in the customer request form.
- *Sequence dependency* Two tasks that must be performed by different roles in a given sequence create a sequence dependency, e.g., an invoice can go to the cashier for payment only after manager approval. Hence, there is a dependency between approval by the manager and the payment operation.

The system architecture of the authorization and control framework is shown in Figure 1. In the WMS, documents are linked to work baskets while workers interact with the documents through the work baskets assigned to them. The access to and operations on the documents are controlled by the workflow manager using the three techniques: *workflow control tables, sequence constraints,* and *event-based workflow management rules* as discussed below.

We assume that work baskets are assigned to workers according to their role. A basket contains similar documents that must be worked upon by workers performing a specific role. Any worker belonging to that

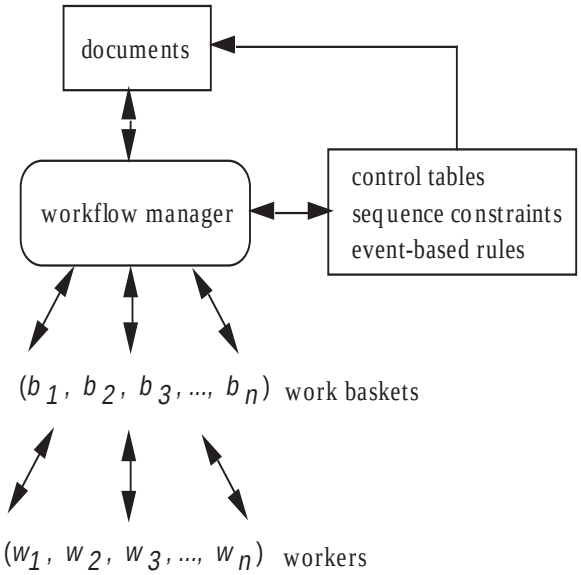

Figure 1: System architecture for the WMS

role can process the documents in the basket. When a worker starts processing a document, the document is "locked" so that no one else can access it.

The operational integrity is maintained by placing restrictions on the kinds of operations (e.g., read, enter, modify, etc.) that a worker in a given role is allowed to perform on fields of a document. This is done by means of workflow **control tables** that specify the permissible operations for a given role. Once a document is accessed by a worker, he or she will be notified of the permissible operations by the system, perhaps using a pop-up menu and other standard graphical user interface features (e.g., a read-only field will be dimmed).

Proper routing of documents to various workers is the most important function of the WMS. In our framework, this is achieved by means of **sequence constraints** that impose dependencies between tasks to be performed on a document. If no sequence constraints are imposed on a document, then it means the various operations can be performed in any random order. This is one extreme situation (which is rather unusual), while the other extreme is that *all* the tasks have to be done in a fixed order. The most common situation, however, lies in between, and this is where our framework can be used to optimize the workflow process and achieve greater efficiency. The proper routing of a document is achieved by controlling the sequence in which a document is accessed by various roles.

Our framework also supports **event-based rules** to enforce additional operational controls that cannot be provided by the above mechanisms. Event-based rules can trigger special actions to take place especially when an unusual event occurs. This is also a useful mechanism for handling exceptions. Some examples of event-based rules are:

- If a document has been in the system for more than 24 hours, then send it to the supervisor immediately (perhaps for special handling).
- If the list of items on the purchase order does not match the list of items on the invoice, then suspend the invoice.
- If the price on the invoice is different from the approved amount, then bypass the other steps and route the invoice directly to the manager, say role 10.
- A document can be deleted after workers w1, w2 and w3 have seen it.

The workflow manager moves the documents among the baskets after each task is completed by consulting the workflow control tables, the sequence constraints, and the event-based rules. It is important to understand clearly the *interaction between the three components* of our framework. The workflow control tables (Section 3) specify absolute, static authorizations that govern rights of access to documents. The sequence constraints (Section 4) specify the temporal order in which documents are accessed, and override the workflow control tables, i.e., a worker (in a given role) may not access a document if the temporal constraints are violated even though he/she may possess the appropriate permissions contained in the workflow control tables. Finally, the event-based rules (Section 5) supersede both the workflow control tables and the sequential constraints, and represent a more sophisticated means of managing the workflow and handling special situations.

3 Workflow Control Tables

In this section, we develop an extensible technique for controlling the flow of documents in a workflow system, and restricting access to documents by assigning permissions and authorizations.

There are two aspects in workflow control: we need a way to limit the operations that an employee can perform to access and modify the document. We call these *read/write operations*. Secondly, we provide operations to monitor and change the status of a document. These are called *status change operations*. Status change operations are required in a workflow system to better track a document through the system, and maintain its integrity. These operations are discussed in more detail next.

Figure 2 shows the various *read/write* and *status change* operations. There are 15 operations in this figure. They have been organized into a tree such that the scope of the permissions or authorizations decreases at lower levels of the tree. Moreover, a permission at a higher node in the tree subsumes (or includes) all permissions at nodes below it. The meanings of the various operations are as follows. The *initialize* operation sets the value in a field to a certain default value. The *enter* operation corresponds to making an entry into an empty (or null) field. *Delete* and *read* operations have their usual meanings. The *increase* and *decrease* operations apply only to numeric fields, and are more restrictive than the *read & write* operation, which modifies a field unconditionally. The *approve* operation is the equivalent of signing

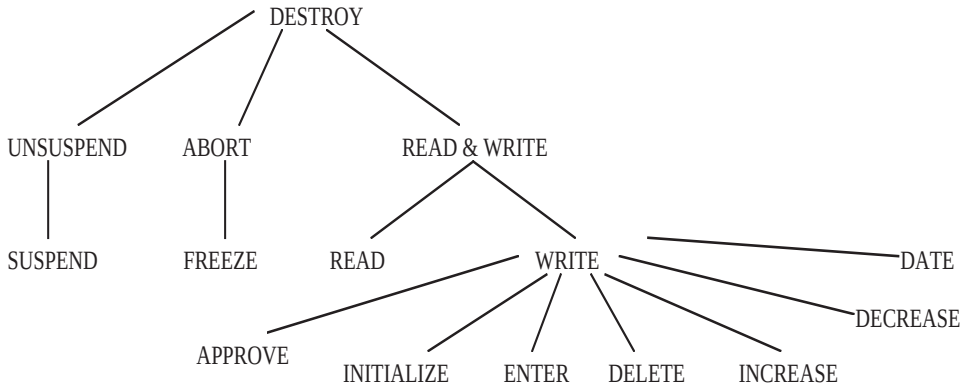

Figure 2: Hierarchy of read/write and status change operations

a document to indicate approval. The *date* operation is similar to putting a time stamp on the document. The time stamp identifies when a certain approval was given or an entry was made.

The status change operations, i.e., *suspend, unsuspend, abort*, and *freeze*, are unique to workflow applications and apply at the document level. These operations are also shown in Figure 2 arranged in a hierarchy such that operations at higher nodes subsume all the lower ones below them in the subtree. Figure 3 shows how these various operations change the state of a document, and what state transitions are permissible.

The descriptions of the operations and the new states that result from performing them are as follows. A document is initially in the *normal* state when it is created. The *freeze* operation moves the document into the *frozen* state, and in this state further modifications to the document are prevented; only read operations can be performed after the freeze. The *suspend* operation puts the document in a *suspended* state. This could occur, for instance, when a worker is unable to process a document or finds an exception in it. Putting a document in a suspended state is a way to attract attention to a problem situation and initiate special handling. Once the problem is resolved, the *unsuspend* operation releases the document from the "suspended" state. The *abort* operation cancels or nullifies a document, but it still exists in the system (perhaps for audit purposes). This operation takes the document into the *aborted* state. In contrast, the *destroy* operation removes the document completely from the system, and is the electronic equivalent of tearing up a piece of paper. The *destroy* operation takes the document into the *destroyed* state. Figure 3 also shows the various read/write operations that may be performed in these different states by associating the notation R (for read) and W (for write) with each state in parentheses. For example, in the normal states both read (R) and write (W) operations are possible, while in the *frozen* and *suspended* states only read operations can be performed. In the other states none of the operations can be performed.

To store this information we envisage two tables or relations, one for status change operations and the other for read/write operations. The first table would have three columns to indicate the role, document and permission. The second table would have four columns to indicate role, document, field and permission.

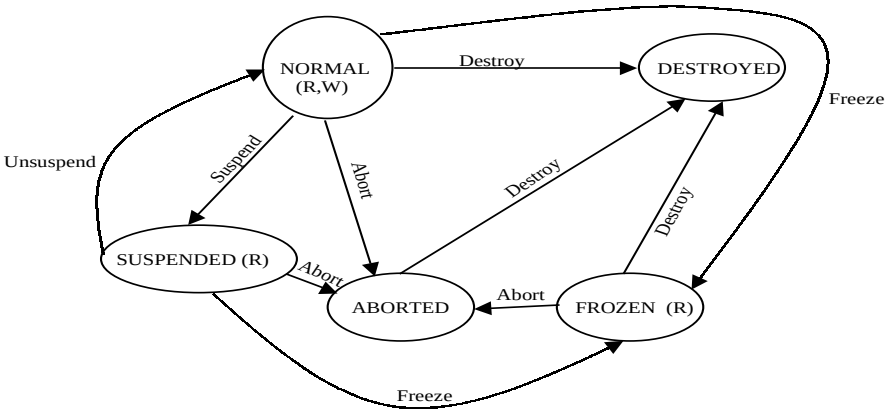

Figure 3: State change graph for status change operations

The schema of the tables is as follows:

PERM1(Role, Doc., Permission)

PERM2(Role, Doc., Field, Permission)

Notice that these tables do not contain any temporal information because they are static and define absolute permissions of access to documents. The next example illustrates how these tables are used.

Example 1 As an example, consider a simple workflow application where an employee (role *traveler*) fills in a travel authorization form with fields for *reason for travel*, *AmountRequested*, etc. The form then goes to an employee in the role *clerk* who fills in the *AmountSpent* field, i.e. the amount spent on previous travel in the year. Finally, a *manager* reviews the form. Initially, the *ApprovedAmount* field has the same value as the *AmountRequested* field. The manager has the discretion to change the *ApprovedAmount*. After the manager's approval, the form is *frozen*, and is available for read-only access to the traveler, clerk and the manager. The PERM1 and PERM2 relations are shown in Tables 1 and 2. Table 1 shows the various permissions for the roles. Each permission subsumes other permissions as per the hierarchy of Figure 2. Table 2 shows the various read/write operations that the roles may perform. For instance, all three roles may perform the *date* and *read* operation on the document, while other operations are restricted. The traveler may make an entry into the *AmountRequested* field; the clerk and the manager may only read this field but not change it. Only the manager is allowed to enter a value into the *AmountApproved* field. Finally, a symbol * in a column is a special string which indicates that this permission entry refers to *all* legitimate values in that column, and it can apply to roles, documents and fields.

Role	Document	Permission
Clerk	TravelDoc	suspend
Manager	TravelDoc	unsuspend
Manager	TravelDoc	abort

Table 1: PERM1 Table for status change operations

The overhead of maintaining these two tables is very small. For instance, consider that, in a large organization, there are 1000 types of documents and 50 roles per document, i.e., individuals in 50 different roles handle the document. Assuming 3 permission entries per role, there would be 150,000 rows in the PERM1 table. In the PERM2 table, we assume there are 10 entries for every document-role-field combination and this would result in a total of 500,000 rows. Assuming 10 characters per field, the size of the PERM1 table would be 4.5 MB, and the PERM2 table would be 20 MB. By suitable encoding of the various fields, these sizes can be further reduced by a factor of 3 or 4. Hence, these are small tables by database standards, and they can be accessed efficiently by indexing PERM1 on role and document combination, and PERM2 on role, document and field.

Moreover, it should be noted that in this way it possible to go down to a very fine degree of detail in the access control specifications for the workflow. It should also be noted that this permission hierarchy has been developed based on needs of a workflow system. The objective of this hierarchy is to permit operations that arise in workflow environments, and yet not be restrictive. For instance, if the permission level is RW (read and write) it gives unconditional permission to read and to make changes. On the other hand, it is also possible to restrict permission very narrowly to only timestamping a document to indicate it has been received. Therefore, this hierarchy is *complete* in the sense that it does not exclude any operation from being performed; however, it is not exhaustive because it is not possible to anticipate all possible operations that may arise. Nevertheless, and most importantly, it is extensible in that it is possible to add other operations to it. For example, other instances of specialized operations are signing a check, issuing a contract or an appointment, scheduling a meeting, etc. These can be added to the hierarchy. Other ways of extending the hierarchy would be, for instance, by further specializing the decrease operation into "decrease by 1000 or less", and "decrease by more than 1000."

4 Workflow Sequence Constraints

4.1 Basic concepts of Sequence Constraints

A *routing scheme* is the specification of routing paths that a workflow is required to follow. A routing scheme can be either *mandatory*, i.e., all steps must be followed in a strict, prespecified order, or it may be *flexible*, i.e., several alternative paths are permitted provided the sequence constraints are not violated. In

Role	Doc.	Field	Permission
*	TravelDoc	*	date
*	TravelDoc	*	read
Employee	TravelDoc	AmountRequested	enter
Clerk	TravelDoc	AmountSpent	enter
Manager	TravelDoc	AmountApproved	enter
Manager	TravelDoc	*	approve

Table 2: PERM2 Table for read/write operations

this paper, we are interested in *flexible routing schemes*. Other approaches for defining such routing schemes are Petri Nets [31] and Information Control Nets (ICN) [14]. Sequence constraints (also known as precedence constraints) occur in many scheduling problems such as mechanical assembly [20, 33], task sequencing in robotic systems [6], machine shop planning [9], and data flow analysis [32]. However, as already discussed in Section 1 there are important differences between these applications and the workflow problem.

A sequence constraint specifies the rules that a document routing should observe so that the business procedures are not violated. When the workflow system supports flexible routing schemes, a set of sequence constraints can be used to define a partial ordering on the routing paths, and the workflow management system would enforce those constraints. A routing path is considered *legitimate* or *permissible* if none of the sequence constraints is violated.

In this paper, our research objective is to support flexible routing by providing a routing scheme consisting of a set of sequence constraints to describe a business process. Therefore, we need a mechanism whereby a set of sequence constraints can be used to define a partial ordering on the document routing, and the workflow management system would serve to enforce those constraints. This thrust of our research distinguishes it from previous research efforts in other application areas discussed above.

4.2 A Process Constraint Language (PCL)

A routing process can be described by *events*, *precedences* (the sequential relationship between events) and *clusters* (a collection of associated events). More formally, they are defined as follows.

Definition 1 . Event (e_i): *An event is any occurrence or action that is considered relevant in a process involving routing of a workflow.*

Some example events are: entering data on a form, modifying the value of a field, approving a document, timestamping a document, etc.

Two events, **origin** and **freeze**, are *special* events that mark the start and the end of a workflow sequence.

Definition 2 . Precedences: *There are four types of precedences in this routing language. These precedences express various kinds of sequential relationships between events in a workflow.*

- **Follow Once ($e_i \rightarrow e_j$):** *This constraint means that at least* `one` e_j *occurrence must appear after one or more occurrences of* e_i *in the routing history.*

- **Follow Each ($e_i \stackrel{e}{\rightarrow} e_j$):** *This constraint requires that* `each` *occurrence of* e_i *must be followed by an occurrence of* e_j*.*

- **Follow Immediate ($e_i \stackrel{i}{\rightarrow} e_j$):** *The constraint states that every occurrence of* e_i *must be followed* `immediately` *by* e_j*.*

- **Follow Not ($e_i \not\rightarrow e_j$):** *This constraint requires that* e_i *must* `not` *be followed by* e_j *in the routing history.*

A *routing history* is a sequential trace of all the events that happened to a document. This sequence may be stored in an array or a queue, where each entry denotes one event. The notation $(e_i \rightarrow e_j \rightarrow e_k)$ is a short-hand notation for representing two separate constraints $e_i \rightarrow e_j$ and $e_j \rightarrow e_k$.

Another important concept involves relationships or associations among several events, and we introduce the notion of *clusters* to treat such situations.

Definition 3 . Clusters. *There are two main types of clusters:*

- **PARA (parallel) Cluster ($\{e_1, e_2, ..., e_n\}^m, 1 \leq m \leq n,\ n \geq 2$):** *A collection of n events, only and exactly any m of which must be executed independently (in any order). This means that no event is required to wait for another event to complete.*

  The curly brackets denote a PARA cluster. There are two special cases or subtypes of PARA clusters: A *PARA-AND (Parallel-AND) cluster* is a PARA cluster with $m = n$. An *OR cluster* is a PARA cluster with $m = 1$.

- **SEQ (sequential) Cluster** *($[e_1, e_2, ..., e_n]^m, 1 \leq m \leq n,\ n \geq 2$): A collection of n events, only and exactly any m of which must be executed in any serial order. That is, no event except for the first one can execute before it sees the result of a prior event.*

  The square brackets denote a SEQ cluster. Again, there is a special case of SEQ clusters: A *SEQ-AND (Sequential-AND) cluster* is a SEQ cluster with $m = n$, i.e., all the events must be executed.

The following two examples illustrate the PCL language.

Example 2 Consider a rule that: any increase to the authorized budget made by the traveler must be followed by a subsequent written approval from the manager. This can be stated as:

`Traveler.budget.increase` $\rightarrow$ `Manager.budget.approve`

This semantics implies that if the traveler increases the budget amount, then approval from the manager is required, but if the traveler *decreases* the amount, then no subsequent approval from the manager is necessary. More complex rules can be expressed using the techniques in Section 5.

Example 3 Consider an operation sequence in which a secretary completes a form *Doc*, and the department chair and deputy chair sign it immediately afterwards, but in any order. This can be stated as:

$$\texttt{Secretary.complete} \xrightarrow{i} [Chair.approve, DeputyChair.approve]^2$$

Note that the cluster in this sequence constraint is a SEQ cluster and follows immediately after the operation on the left. However, the approvals can be done in any order.

The following examples illustrate *complete* and *partial* orderings.

Example 4 Consider the following set of constraints, where r_i's are roles that access a document: origin $\xrightarrow{i}$ r1, r1 $\xrightarrow{i}$ r2, r2 $\xrightarrow{i}$ r3, r3 $\xrightarrow{i}$ r4, r4 $\xrightarrow{i}$ freeze. Under this set, there is only one possible document history: r1,r2,r3,r4,freeze. Thus, this is a complete ordering.

Example 5 Now, consider the following set of constraints:

$$\text{origin} \xrightarrow{i} \text{r1},\ \text{r1} \rightarrow [r2, r3]^2 \rightarrow \text{r4},\ \text{r4} \xrightarrow{i} \text{freeze}.$$

Under this set, possible legitimate routing histories are:

origin,r1,r2,r3,r4,freeze, and

origin,r1,r3,r2,r4,freeze.

This is an example of a *partial ordering*. It is very useful when two steps, such as $r2$ and $r3$ above, must be performed in sequence, but not in any specific order. Similarly, a sequential cluster can be used to express the requirement that any two out of three vice-presidents (VPs) in the company must sign a check *in sequence* to release a payment, as follows:

$$e_x \rightarrow [vp1, vp2, vp3]^2 \rightarrow e_y$$

where $vp1, vp2, vp3$ are events corresponding to the three Vice-President approvals, and e_x and e_y are some other events.

4.3 Completeness of the PCL Language

Next, we show the completeness of PCL by using it to describe Petri Nets [31], a well recognized, graphical language for representing processes. We show that PCL is "complete" because it has the complete functionality of Petri Nets which is a proven language.

A Petri Net (PN) has three types of elements: *places* (P), *transitions* (T), and *state tokens* (S). Informally, a place in a PN is equivalent to an event in the PCL, and a transition in a PN is equivalent to an immediate precedence in the PCL. However, a state token in PN does not require a counterpart in PCL.

According to the Workflow Management Coalition [43], there are six workflow primitives: AND-join, AND-split, OR-join, OR-split, Iteration, and Causality. *These six workflow primitives can be used to model any routing scheme.* Petri Nets can be used to model the six workflow primitives [41] as shown in Figure 4.

To prove Theorem 1, we need only to show that there is a PCL constraint with respect to each Petri Net (PN) primitive. Next, we prove six lemmas corresponding to the six primitives, respectively.

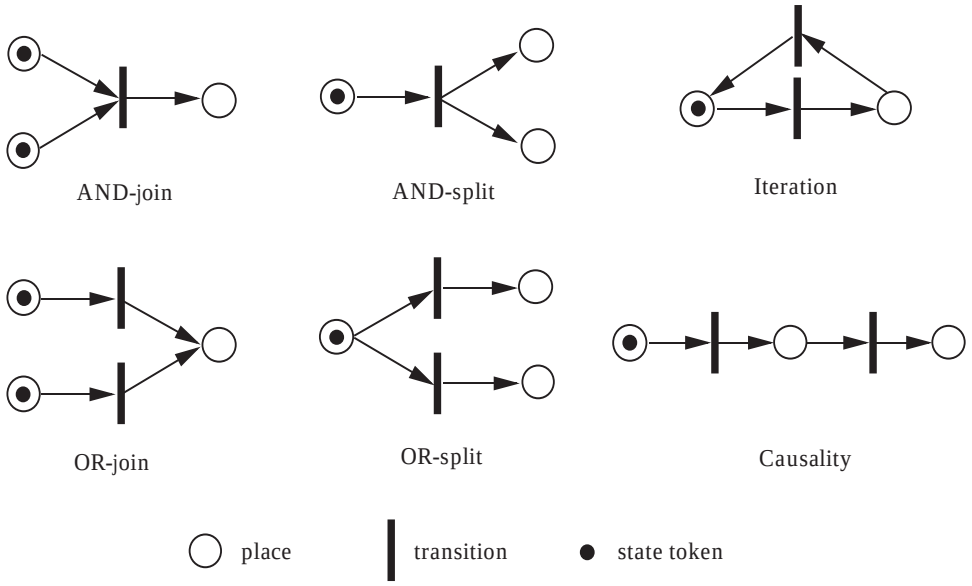

Figure 4: Workflow primitives

Lemma 1 *For each AND-join in a PN, there is a corresponding PAR-AND Cluster in the PCL.*

Proof: Given a PN containing an AND-join, where p_1 and p_2 are the origin places and p_3 is the target place, the AND-join can be expressed by a PCL constraint consisting of a PAR-AND cluster, which appears at the head of an immediate Precedence constraint as:

$$\{p_1, p_2\}^2 \xrightarrow{i} p_3.$$

□

Lemma 2 *For each AND-split in a PN, there is a corresponding PAR-AND Cluster in the PCL.*

Proof: Similar to above, but now the PAR-AND cluster appears at the tail of the constraint:
$p_1 \xrightarrow{i} \{p_2, p_3\}^2$. □

Lemma 3 *For each OR-join in a PN, there is a corresponding OR Cluster in the PCL.*

Proof: Given a PN containing an OR-join, where p_1 and p_2 are the origin places and p_3 is the target place, the OR-join can be expressed by a PCL constraint consisting of an OR Cluster and an Immediate Precedence:

$$\{p_1, p_2\}^1 \xrightarrow{i} p_3.$$ □

Lemma 4 *For each OR-split in a PN, there is a corresponding OR Cluster in the PCL.*

Proof: Similar to above, but now the OR cluster appears on the right hand side of the constraint as:
$p_1 \xrightarrow{i} \{p_2, p_3\}^1$. □

Lemma 5 *For each Iteration in a PN, there is a corresponding cyclic sequence of Immediate precedence constraints in the PCL.*

Proof: Given a PN containing an *iteration* where p_1 is one place and p_2 is the other place, this *iteration* can be expressed by two PCL constraints consisting of two *immediate precedences*:

$p_1 \stackrel{i}{\rightarrow} p_2$, $p_2 \stackrel{i}{\rightarrow} p_1$. □

Lemma 6 *For each causality in a PN, there is a chain of two immediate precedence constraints in the PCL.*

Proof: Given a PN containing a causality where p_1 is the origin place, p_2 is the middle place and p_3 is the final place, the causality can be expressed by two PCL constraints consisting of two immediate precedences:

$p_1 \stackrel{i}{\rightarrow} p_2$, $p_2 \stackrel{i}{\rightarrow} p_3$. □

Now, a theorem can be stated as follow:

Theorem 1 *For each Petri Net, there is a set of PCL constraints that uniquely determines the Petri Net.*

Proof: Any PN can be seen as a collection of PN primitives by definition. If each PN primitive can be represented by a PCL constraint, then the whole PN can be defined by the collection of these PCL constraints. From Lemmas 1 through 6, all six PN primitives can be defined by PCL constraints. Therefore, Theorem 1 follows. □

In summary, we have shown PCL to have at least the same modeling capabilities as Petri nets. In addition, PCL contains other features that Petri Nets do not have for defining constraints in terms of various sequential precedences and clusters. For instance, the constraints in Examples 2, 3, and 5 are difficult to express using Petri Nets, and in this respect Petri Nets clearly offer less flexibility.

Moreover, the philosophy of PCL is to only represent the constraints that apply to the routing scheme, and permit everything else that is not specifically prohibited. This makes it easier to describe flexible workflows where there are few restrictions. However, Petri Nets are difficult to work with in such loosely defined situations. As a result, we can conclude that PCL provides richer expression capability than Petri Nets, and therefore PCL is better suited for modeling flexible routing schemes.

On the other hand, Petri Nets are superior when a graphical representation of the model is required. PCL does not have a graphical representation. Although some features of PCL can be represented graphically, the various precedences such as Follow Not and the SEQ cluster are difficult to represent graphically.

4.4 Constraint Verification and Enforcement

The WMS must enforce the partial ordering defined by sequence constraints and refuse routing paths that are not legitimate. There are two steps in this enforcement. First, the WMS must reject any inconsistent set of constraints at scheme specification time. Second, the WMS must reject any attempt to route a document in a way that would create a nonpermissible history. This is done at execution time when the document is being routed.

4.4.1 Verification

To verify the routing scheme, we need to check for:

- *redundant* constraints
- *inconsistent* constraints
- *incomplete* constraints, and
- *syntactic errors* in constraint specifications.

We envision a GUI interface consisting of icons (or hot buttons) corresponding to the PCL language elements such as events, precedences and clusters for defining the routing scheme. The following rules are used for verification of sequence constraints.

1. The set of constraints should not contain any duplicate constraints. In case of two or more overlapping constraints, the more restrictive constraint will dominate.
2. Given any two events e_i and e_j, $i \neq j$, there should be at most one of the following constraints: $e_i \rightarrow e_j$, $e_i \xrightarrow{i} e_j$, $e_i \xrightarrow{e} e_j$, $e_i \not\rightarrow e_j$.
3. Both, $e_i \xrightarrow{i} e_j$, and $e_i \xrightarrow{i} e_k$, should not appear together in the specification for a document although $e_i \xrightarrow{i} [e_j, e_k]^2$ is permitted.
4. Both, $e_i \xrightarrow{i} e_k$, and $e_j \xrightarrow{i} e_k$, should not appear together in the specification although $[e_i, e_j]^2 \xrightarrow{i} e_k$ is permitted.
5. For a set of sequence constraints, there should be a constraint in the form of $E_i \xrightarrow{i} freeze$, where E_i is either a single event or an event cluster.
6. For a set of sequence constraints, there should be a constraint in the form of $origin \xrightarrow{i} E_i$, where E_i is either a single event or an event cluster.
7. Every constraint must be complete, i.e., there must be a head, a tail, and a precedence arrow. The head or tail may either be a single event or an event cluster.
8. The head and tail must not be identical.

Rules 1 and 2 ensure that there is no redundancy in the set, Rules 3 and 4 remove any inconsistencies, Rules 5 and 6 check for completeness, and Rules 7 and 8 are used to eliminate syntactical errors. Note that both $e_i \xrightarrow{i} e_j$ and $e_j \xrightarrow{i} e_i$ can occur in a constraint set since iteration is allowed.

The overhead of constraint verification is minimal as constraints are usually modified relatively infrequently in a workflow management system. Therefore, we omit the efficiency analysis for constraint verification.

4.4.2 Enforcement

Assuming that the routing scheme has been defined and checked according to the above rules, the next step is to enforce the constraints at the time of workflow execution. Consequently, whenever a routing request is issued at execution time, the WMS must check that none of the constraints will be violated by accepting the request. If a violation results, then the request must be refused. Moreover, when a request is made to *freeze* a document, the WMS must check to ensure that no constraints would be violated by accepting the freeze.

To develop the constraint enforcement rules, we first need to define some basic concepts related to instantiation and activation of events and constraints:

- *Event instantiation.* When an executed event is found in the routing history of the document, the event is said to be *instantiated*. When the events necessary to completely execute a PARA or SEQ cluster are found in the routing history of the document, the cluster is *instantiated*. A list is maintained to keep track of instantiated events and clusters.

- *Constraint activation.* A constraint is activated when the necessary events associated with its head are instantiated. In general, the head of a constraint can be a single event, a PARA cluster of events (including both AND cluster and OR cluster as special cases) or a SEQ cluster. A PARA cluster or a SEQ cluster is activated with multiple events. We assume that a list of activated constraints is maintained in a suitable data structure.

- *Constraint deactivation.* A Follow Immediate constraint is deactivated when its tail is instantiated. This is done because once the constraint has been satisfied, it becomes obsolete and should be removed from the set of active constraints. Similarly, two additional types of constraints, namely Follow Once and Follow Each, must also be subject to deactivation once their tails are instantiated and they are satisfied as a result. Follow Not constraints need not be deactivated because they should always be enforced once activated.

Next, we present the rules for constraint enforcement each time a request is made to carry out an event.

1. If the requested event violates any Follow Immediate constraint (i.e., it is Not in the tail of the active Follow Immediate constraint, then we deny the request.

2. If the requested event violates any Follow Not constraints (i.e., it is in the tail of a Follow Not constraint), then deny the request.

3. If the requested event is "freeze", we must prevent immature "freeze" of the document. Therefore, we need to check if there is any active Follow Once and Follow Each constraint whose tail has not been instantiated. If this is the case, the enforcement algorithm should refuse the "freeze" request.

The overhead of constraint enforcement is proportional to the number of instantiated events (N_E). The order of computational complexity for each event is proportional to the number of active constraints (N_C) defined on the type of document the event accesses. If the event is "freeze", the order of computation for the algorithm is $\emptyset(N_{fo} + N_{fe})$ where N_{fo} and N_{fe} are the number of active Follow Once and Follow Each constraints. For all other types of events, the order of computation for the algorithm is $\emptyset(N_{fi} + N_{fn})$ where N_{fi} and N_{fn} are the number of active Follow Immediate and Follow Not constraints. Consequently, the computational complexity of the algorithm is $\emptyset(N_E N_C)$.

5 Event-Based Workflow Management Rules

5.1 Introduction

The third component of the workflow management framework consists of event-based workflow management rules. The workflow control tables specify the operations each role may perform on a document or field, and the set of sequence constraints determine the permissible sequence in which they may be performed. However, there are several types of controls and operations that cannot be supported by workflow control tables and sequence constraints because they cannot express events and conditions. Event-based rules can be employed in several ways.

1. *Routing.* Event-based rules can be used to support sophisticated routing based on complex conditions.
2. *Monitoring.* The quality and efficiency of operations can be monitored by event-based rules for managerial purposes.
3. *Control Rules.* Such rules may be used to prohibit unauthorized operations.
4. *Operations Rules.* These rules may be used to carry out automatic operations when specific conditions are satisfied.
5. *Exception Handling Rules.* These rules are useful for special situations, such as granting temporary authorizations and for special handling of documents.

We distinguish two types of data elements in the document: *fields* and *attributes*. A *field* is a data element that is displayed to the user. Field values are entered and updated by end users. An *attribute*, on the other hand, is a data element or metafield that describes the features of the document and is usually *hidden* from the user. For instance, a document should have an *identifier*, a *timestamp* specifying the time it was created, its *type* such as invoice, travel authorization, and loan application, and a *status*. Document attributes are usually maintained by the system.

We extend the Event Condition-Action (ECA) rules in active databases [27, 19, 22] into Event-Role-Object-Condition-Action (EROCA) rules with the addition of the *role* and *object* clauses. The BY clause

specifies the name of the role operating on the document, and the TO clause specifies the object (i.e., a document or a field of a document) in question. These two clauses are necessary in workflow management because authorizations and controls require the knowledge of the roles and the documents. They are not needed in a database environment because the authorizations and operations in databases are relatively simple and the data elements are not document dependent.

RULE	*id*
ON	*event*
BY	*role*
TO	*object*
IF	*condition(s)*
THEN	*action(s)*

Each rule is identified by its *id*. An *event* is specified in the ON clause, and is an occurrence of an operation such as read, enter, initialize, abort, decrease, etc. As discussed later, system events are also related to the document status, e.g., an event may be triggered upon *submission of a completed task* by a role to the WMS.

A *role* in the BY clause can be any role or a set of roles. They must be predefined in the workflow management system so that they can be uniquely identified by the rule. A role set acts as a short hand for the collection of roles to simplify the rule definition.

An *object* in the TO clause specifies the document (or type of document) to which the rule belongs and can be represented as a combination of document, attribute and field using the dot syntax. Valid object expressions are therefore, *document, attribute, field, document.attribute* and *document.field.*

A *condition* in the IF clause is comprised of logical literals combined by AND and OR operators. Each literal has the form: `operand1 <op> operand2`. A valid operand can be *Doc.Attribute* or *Doc.Field* or constant. An <op> can be any of the elementary logical operator such as $<$, $>$, $\leq$, $\geq$ and $==$.

Actions in the THEN clause can be either system-defined operations or routing actions. Furthermore, multiple actions may be defined in the same rule if they pertain to the same event and the same conditions.

Note that the BY, TO and IF clauses are optional. A missing clause would result in a more general rule. For instance, when the BY clause is omitted in a rule, the rule would apply to all roles in the system. If the BY and TO clauses are missing in the rule, it is then a systemwide rule that applies to all documents and roles.

Next, we discuss and give examples for the five types of workflow management rules listed above. As we proceed, we shall explain various new functions and events that are summarized in Table 3.

5.2 Routing Rules

Event-based rules can be used conveniently to define routine routing. For instance, let $Form$ be a document that is filled in by role $r1$, and must be verified by role $r4$. This routing can be specified by the following rule.

RULE 5.1
ON Done
BY r1
TO Form
IF Form.Status == "Filled"
THEN ROUTE(r1,r4,Form).

Note that the *Done* event is generated by the system when role $r1$ completes a task, and the routing is performed automatically by the system. ROUTE(r1,r4,Form) redirects the document called $Form$ from $r1$ to $r4$. More complex routing can also be performed as follows.

RULE 5.2
ON Done
BY r4
TO Form
IF Form.Status == "Verified_Correct"
THEN ROUTE(r4,r5,Form)
ELSE ROUTE(r4,r3,Form).

This rule directs the document $Form$ to either role r3 or role r5, depending on its status after r4 completes work on it. As this example illustrates, routing rules provide support for normal routing, and also enable special routing in certain situations. Thus, routing rules complement sequence constraints by providing additional semantics and by allowing selection of a specific route when the sequence constraints permit multiple alternatives.

5.3 Monitoring Rules

RULE 5.3
ON Checkup
TO Doc
IF Doc.Status $\neq$ "Completed"
AND DAYS(Doc) $>$ 10
THEN ROUTE(*,Manager,Doc)
AND MESSAGE(Manager,Doc,"Document age is over 10 days.").

The above rule routes an *incomplete* document, more than ten days old, to the Manager role. The term *Status* is a document attribute which takes values such as "Incomplete", "Completed," etc. $DAYS()$ is a system function that returns the age of the document in number of days. The *Checkup* event is a system operation that occurs periodically and performs operational control functions to ensure certain business policies are enforced.

5.4 Control Rules

The following workflow management rules exercise control on document operations. Among the next three rules, the first rule enforces a business requirement that Class 3 documents, less than seven years old, should not be destroyed and the next two are data integrity rules.

RULE 5.4
ON Destroy
TO Doc
IF Doc.Class == 3
AND YEARS(Doc) < 7
THEN REJECT.

In the above rule, note that the term Class is a document attribute and the function YEARS() is a system function that extracts the age of a document in years.

RULE 5.5
ON Update
TO Doc.F1
IF Doc.F1 > 10,000
THEN REJECT.

This rule rejects any value over 10,000 assigned to field F1.

RULE 5.6
ON Increase
TO Doc.F1
IF NEW(Doc.F1) − OLD(Doc.F1) > 1,000
THEN REJECT.

This rule disallows any increment to field F1 of over 1,000. Note that the functions NEW and OLD are system functions that indicate the new and old values of the field.

5.5 Operations Rules

Certain operations on documents may be automated. That is, prescribed actions will be taken by the system upon the occurrence of a given event and the satisfaction of specified conditions.

RULE 5.7
ON Done
BY r3
TO Doc
IF Doc.F1 == "Approved"
THEN Doc.Status = "Closed".

RULE 5.8
ON Cleanup
TO Doc
IF Doc.Status == "Closed"
AND DAYS(Doc) ≥ 30
THEN DESTROY(Doc).

When document *Doc* is approved, the first rule above sets the *Status* attribute of the document to "Closed" automatically. The second rule automatically destroys document *Doc* after it has been closed if its age is over 30 days. The *Cleanup* event is a system event which occurs periodically to remove obsolete files.

Name	Description
ACCEPT	system action to permit a user override operation.
Checkup	system event, occurs periodically for operational control.
Cleanup	system event, occurs when the system removes obsolete files.
DAYS(Doc)	system function call to return the document age in days.
DESTROY(Doc)	system action to destroy the document Doc.
Done	system event, occurs when a user operation is done.
MESSAGE(Role,Doc,"msg")	system action to alert Role about Doc with "msg".
REJECT	system action to reject a user operation.
ROUTE(r1,r2,Doc)	system action to route Doc from r1 to r2.
YEARS(Doc)	system function call to return the document age in years.

Table 3: Action, function, and event verbs for the WMS

5.6 Exception Handling Rules

Consider an example where, per the workflow permission tables, $r4$ is normally allowed to *abort* document *Doc*, but $r5$ is not. What happens when all workers in role $r4$ are absent?

RULE 5.9
ON Abort
BY r5
TO Doc
IF r4.Status == "Absent"
THEN ACCEPT.

This rule states that role r5 is allowed to abort document *Doc* if role r4 is absent. *Status* is an attribute of a role, and is expressed in this case as *r4.Status*.

A canonical list of various verbs for events and actions is given in Table 3. Several events were used in the workflow management rules including abort, destroy, increase, update, checkup, cleanup and done. The first four of these events are user-initiated workflow operations (described in Section 3), while the last three events (i.e, cleanup, checkup, and done) are system-initiated events. Several additional verbs are used in the action clauses of the rules, such as ACCEPT, DESTROY, MESSAGE, REJECT, and ROUTE.

5.7 Rule Management Issues

An industrial strength workflow management system will have a large number of such rules, and there are important rule management issues that arise in this context and must be addressed. Basically, the rule management system must include components for:

- Ensuring consistency of rules and disallowing non-functional rules.
- Indexing of rules.

- Conflict resolution among multiple candidate rules.

We expect that these rules will be stored in a database and indexed appropriately for efficient retrieval at run-time. Most current database systems from leading vendors such as Oracle, Sybase, Microsoft and Informix provide considerable support for active rules in addition to integrity constraints. For example, one can write rules such as: When a 10% raise is given to Mike, also give a 10% raise to Betty. The collection of rules is indexed in such a way that when an update is performed on Mike's salary, this rule is triggered. An additional rule which would give Mike a 10% raise every time Betty got one, would create a non-functional cascading situation, which is detected and disallowed. Techniques for indexing rules in databases are discussed in [19, 3, 36].

Another important issue relates to *conflict resolution* among competing rules. These are situations in which multiple rules can become eligible for "firing." In such situations, the workflow system must decide and select one rule that should fire at a time. This is done by assigning priorities to rules, or by some other method such as specificity of rules. These situations arise frequently in expert systems, e.g., OPS5 [8]. More recently, researchers have developed techniques for handling them in database systems by borrowing and extending some ideas from expert systems research. Techniques for conflict resolution in database systems are discussed in [37, 23, 29, 30]. We anticipate that an implementation of our rules framework will draw on existing research on management of large numbers of rules in database systems, and extend it further by adopting it to the new structure of the rules.

6 A Case of Workflow Management in Consumer Loans

We now illustrate how the proposed framework can be used in an application of consumer loan management. Consider a major bank that manages its consumer loans through its consumer finance group (CFG). CFG has over 300 office workers (including managers) who process thousands of loan applications every week. Each consumer loan falls into one of over 100 types such as automobile loans, boat loans, home improvement loans, etc. Presently, the loan approval process requires the transmission of documents between the branches and CFG through faxes, and this process results in many redundant copies of the same document. Because the quality of the fax transmission varies, many documents are not entirely readable, resulting in errors, excessive communications and delays. The average turn-around time for consumer loans is more than two weeks which is too long by current standards. Consequently, in roughly 50% of all loan applications, consumers run out of patience and find alternatives in other banks even before CFG finishes the approval process.

The bank is currently implementing a client-server system to transfer data electronically so that fax and telephone calls will be reduced to a minimum. The goal of CFG is to reduce the average loan approval time by a week. However, CFG also realizes that a workflow management system is needed to improve efficiency of workflow in the office.

A simplified description of the loan approval process is described next. The main tasks in the workflow are listed below along with the associated roles.

1. A customer applies for a loan at the loan desk in a branch.

2. (role r0) One of the three workers (w1, w2, and w3) enters customer data into the application form; the collection of tasks performed by these workers defines role r0.

3. The application form is then sent to CFG.

4. (role r1) The reception desk (computerized or manual) at CFG classifies the loan application.

5. (role r2) Workers w4 and w5 check the application forms to ensure that all the data is in place. (Note that r1 and r2 *can* occur in parallel.)

6. If the data is incomplete, then the form is sent back to the branch (role r0) for rework. Otherwise, the form is forwarded in parallel to three groups of specialists.

7. (role r3) Workers w6, w7 and w8 validate the income and assets data.

8. (role r4) Workers w9 and w10 check the credit history of the client.

9. (role r5) Workers w11, w12 and w13 check to see if the client has any outstanding loans.
   (Note that the work by roles r3, r4 and r5 can be done in parallel.)

10. (role r6) One of the managers (w14 or w15) makes a decision on whether the loan should be approved based on the information on the income and assets data and the other outstanding loans. The manager also suggests the amount of the loan.

11. (role r7) One of the senior managers (w16 or w17) makes a decision on whether the loan should be approved based on the credit history, and also determines the amount of the loan. (Note that r6 and r7 should *not* be done in parallel to prevent conflicting decisions.)

12. (role r8) The loan is then sent to the underwriter (w18) for final approval.

13. The loan application is sent back to the branch, and the branch informs the customer of the result. The customer then decides whether to accept the loan offer (if it is approved), and whether to appeal the decision (if the loan is disapproved).

14. (role r9) If the customer decides to assume the loan, then CFG will have one of the managers (w19 or w20) finalize the loan agreement document.

15. At any point in time, the customer may withdraw the loan application. The branch then informs CFG to abort the loan approval process.

Role	Document	Permissions
r0	Loan Application	Suspend
r1	Loan Application	Suspend
r1	Loan Application	freeze
r2	Loan Application	Suspend
r2	Loan Application	freeze
r6	Loan Application	Destroy
r7	Loan Application	Destroy
r9	Loan Application	Suspend

Table 4: The document-level workflow control table PERM1

Applying our framework to the loan approval process, the document-level workflow control table for roles r1 through r9 is shown in Table 4. It shows that r0, r1, r2 and r9 can *suspend* an application; r1 and r2 can *freeze* it; and r6 and r7 can *destroy* it. (In reality, a loan application would not be destroyed upon rejection, because most banks maintain such records for several years.) Furthermore, it should be noted that *destroy* subsumes all other status change operations such as suspend, abort, freeze, and unsuspend (see Figure 2).

Similarly, a field-level workflow control table PERM2 can also be created to store read/write permissions for each field of the document as discussed in Section 3. We will omit PERM2 here for brevity.

The sequence constraints for the loan application workflows are:

1. origin $\stackrel{i}{\rightarrow} r0$
2. $r0 \rightarrow \{r1, r2\}^2$
3. $\{r1, r2\}^2 \rightarrow \{r3, r4, r5\}^3$
4. $\{r3, r4, r5\}^3 \rightarrow [r6, r7]^2$
5. $[r6, r7]^2 \rightarrow$ r8
6. r8 $\stackrel{i}{\rightarrow}$ r9
7. r9 $\stackrel{i}{\rightarrow}$ freeze.

These 7 sequence constraints succinctly summarize the various requirements of the business process. Note that in the above constraints, r1 and r2 can take place in parallel, and so can r3, r4 and r5. On the other hand, r6 and r7 must occur in a (random) sequence as the SEQ cluster denotes.

Rework is common for r0 because sometimes the customer may give inaccurate information, or the customer may decide to change the type of loan or the amount of loan midway through the process. In addition, r1 and r2 may discover entry errors made by r0 and send the document back for corrections. While

rework is unavoidable, excessive rework is undesirable. To keep track of the number of times rework is done, one can add a **monitoring** rule as follows:

RULE 6.1
ON Update
BY r1
TO Doc
IF VISITS(r1, Doc) > 2
THEN MESSAGE(r7,Doc,"Being reworked more than twice by r1.").

The function VISITS(r1,Doc) returns the number of visits to document *Doc* by rule r1.

If, every morning, a manager in role *r6* wishes to find out about all loan applications that are in process for more than ten days, she can use another **monitoring** rule as follows:

RULE 6.2
BY r6
ON Checkup
IF Loan_App.Status $\neq$ "Completed"
AND DAYS(Loan_App) > 10
THEN MESSAGE(Manager,Loan_App,"Document age is over 10 days.").

When a customer decides to withdraw her application, the system should automatically abort the application form and any subsequent updates must be rejected. This can be done using an **operations** rule, assuming that the document has an attribute named *Action*.

RULE 6.3
ON Update
TO Doc
IF Doc.Action == "Withdraw"
THEN Abort(Doc).

Subsequent update attempts will be rejected by the following **control** rule.

RULE 6.4
ON Update
TO Doc
IF Doc.Status == "Aborted"
AND Doc.Action == "Withdraw"
THEN REJECT
AND Message("Document is aborted because the loan application was withdrawn.").

This case illustrates how workflow management can be applied to the loan approval process. Similar applications of workflow management can increase the ability of managers to organize, monitor, control and evaluate business workflows. With a workflow management system, business procedures and policies can be embedded in the automated workflow processes. Thus, a workflow management system can release managers from certain routine and micro-level management activities, and give them more time for making business decisions. One of the key issues in automating business processes is to support flexibility and adaptability [39], and the model proposed in this paper serves this objective by making it easy to perform changes in procedures and policies.

7 Discussion and Comparison with Related Work

We have presented a new approach to workflow design that is different from conventional approaches. It emphasizes *dynamic routing* and is based on enumerating various tasks to be performed, and then imposing a minimal set of sequence constraints on the tasks. This maximizes the alternative routes that a document may take and increases parallelism without sacrificing accuracy. Conventional approaches usually specify a flow pattern by means of a flowchart, and thereby, limit the alternative routing paths. The process of defining constraints in our technique also encourages users to seriously question the need for every constraint.

In Section 3, it was noted that the permissions in the workflow control tables can be defined at various levels of granularity, i.e., for a single field, a group of fields, or even an entire document. Similarly, the sequence constraints of Section 4 and the event-based rules of Section 5 can also apply at different levels of granularity. For instance, a sequence constraint could refer to an operation on a single field, a group of fields or a document. Similarly, the ON field in an EROCA rule can be a document, a single field or a group of fields.

Next, we compare our work with other related work on workflow management systems. Most models of workflow management systems are based on static routing, and usually do not support any sophisticated kind of operational integrity controls. In this respect, our framework is a departure from existing work on workflow systems. Some of the key differences are summarized below.

1. In a static routing approach, routing is explicitly defined using Petri-Nets [44] or Information Control Nets (ICN) [13, 14]. The basic assumption is that the typical routing of tasks does not change much. However, we have relaxed this assumption by imposing sequence constraints and permitting all routing sequences that are not expressly prohibited.

2. The approach for restricting the permissible operations on a document is also a novel part of our model. The absence of such controls means that erroneous operations can occur, often leading to costly rework and inefficiency. We provide such controls through workflow control tables and event-based rules. Another different approach, designed for implementing internal accounting controls, is described in [1, 2]. That approach is based on first-order logic.

3. Since we support flexible sequences of work using the SEQ clusters, it can considerably enhance parallelism because workers may access the same document in any permissible sequence.

4. Static routing has the advantage that it is relatively simple to implement. It is most suitable for environments where there are only a few types of work and the routing schemes of the work are relatively stable. On the other hand, our model is more complex and, therefore, appropriate for an environment that is characterized by many types of work involving flexible routing patterns.

In Section 1 we described how workflow management is different from job shop scheduling. It is also

important to distinguish our work from database management. Although both workflow and database systems deal with business data, workflow management focuses on the management of business processes while database management focuses on the management of data that results from business processes. Naturally, workflow integrity control is much more complex than data integrity control. There are only two data operations concerning data integrity control, namely read and write, while in the workflow control component proposed in Section 3 there are 15 different operations. Data integrity control is implemented by means of simple locking mechanisms that prevent simultaneous write operations to the same data element. Furthermore, operation sequencing as we have seen is a major issue in the design of a workflow management system, but a non-issue in database systems. On the other hand, workflow and database management have a complementary relationship in that the database serves as a data store for the workflow system.

We have covered various implementation issues throughout the paper wherever appropriate. Section 2 describes the high level system architecture. Section 3 describes how the workflow control tables are implemented by means of database relations. In Sections 4.4 and 4.5, we discussed how sequential constraints are verified and enforced. Rule implementation issues were addressed in Section 5.7. Since this paper focuses on defining and justifying the conceptual framework, more detailed discussion of system implementation is deferred to a separate paper.

8 Conclusions

Recently there has been a flurry of research interest in the workflow area on the development of workflow management systems that offer flexible and dynamic mechanisms for routing and operational controls in Workflow Management Systems (WMSs).

This paper has developed a new framework for implementing dynamic routing and operational control in workflow management systems. Our framework is based on three components, which together provide a complete paradigm for managing dynamic workflow applications. The first component is the authorization mechanism, which is based on creating a hierarchy of canonical operations that can be performed by a role. This mechanism is implemented by means of workflow control tables. The second component consists of sequence constraints which impose restrictions on the route of a document and the sequence in which operations are performed. The function of sequence constraints is to allow flexible work sequences by specifying what must be done and what cannot be done. Any workflow pattern that is not explicitly restricted by sequence constraints is considered permissible. The third component is based on event-based workflow management rules that are used to define business procedures and policies in a *declarative* manner. Workflow rules also enable handling of exceptions to normal routing and allow special actions to occur when predetermined conditions are satisfied.

The basic premise of business process reengineering is that business processes must adapt to the constant changes in business environment, both internal and external. This new framework results in a more advanced

workflow management system for complex business applications because it offers both powerful modeling tools and maximum flexibility for managing business workflows. Simple workflow management systems such as Lotus Notes provide little support for routing and operational control beyond database and email applications; therefore, they are inadequate for complex business applications. More advanced systems such as ActionWorkflow and FlowMark do provide routing support based on static routing schemes; however, the drawback of these systems is that they do not offer flexibility for those applications where rework and exceptions arise frequently.

Another innovative contribution of this study is the Process Constraint Language (PCL) presented in Section 4.2. We have shown that PCL provides richer expression capability than Petri Nets, and, therefore, is better suited for modeling flexible routing schemes. On the other hand, PCL does not have a graphical representation for the various language constructs such as the *Follow Not* precedence and the *SEQ cluster*.

We are currently focusing our research efforts on (1) building data structures for developing a prototype system that implements our workflow model; (2) developing efficient algorithms for implementing the techniques; and (3) designing a more user-friendly and English-like specification language for defining and manipulating the three system components.

References

[1] A. Bailey, et. al., "An OIS model for internal accounting control evaluation," ACM Transactions on Office Information Systems Vol. 1, No. 1, January 1983.

[2] A. Bailey, et. al., "TICOM and the analysis of internal controls," The Accounting Review, Vol. LX, No. 2, April 1985.

[3] Brant, D., and D. Miranker, "Index support for rule activation," Proc. 1993 SIGMOD Conference, May 1993.

[4] C. Bussler and S. Jablonski,"Implementing agent coordination for workflow management systems using active database systems," Proceedings of IEEE International Workshop on Research Issues in Data Engineering: Active Databases Systems, Houston, TX, USA, 14-15 Feb. 1994.

[5] C. Bussler and S. Jablonski,"An approach to integrate workflow modeling and organization modeling in an enterprise," Proceedings of the 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, WV, USA, 17-19 April 1994.

[6] T. Cao and A. C. Sanderson, "Task sequence planning using fuzzy Petri Nets", IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, No. 5, May 1995.

[7] M. M. Compton and S. Wolfe, "Intelligent validation and routing of electronic forms in a distributed workflow environment," Proceedings of the Tenth Conference on Artificial Intelligence for Applications, San Antonia, TX, USA, 1-4 March 1994.

[8] Cooper, T., and N. Wogrin, "Rule-based programming with OPS5," Morgan-Kaufman Publishers, 1988.

[9] R.L. Daniels and J.B. Mazzola, "Flow shop scheduling with resource flexibility", Operations Research vol.42, no.3 (May-June 1994) p504-22.

[10] T. Davenport, "Business Process Innovation," HBS Press, 1993.

[11] C. Date, "An Introduction to Database Systems (Volume 1)," Addison Wesley, Fourth Edition, 1986.

[12] W. Du, S. Peterson, and M. C. Shan,"Enterprise workflow architecture," Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 6-10 March 1995.

[13] C. A. Ellis and G. J. Nutt, "Office Information Systems and Computer Science," ACM Computing Surveys, Vol. 12, No. 1, March 1980.

[14] C. A. Ellis and G. J. Nutt, "Modeling and enactment of workflow systems," Proceedings of the 14th International Conference on Applications and Theory of Petri Nets, Chicago, IL, USA, 21-25 June 1993.

[15] C. A. Ellis, personal communication.

[16] F. Fritz and H. Lau,"Workflow requirements for enterprise applications," Proc. IEEE Compcon, March 1994.

[17] J. A. Gulla and O. I. Lindland,"Modeling cooperative work for workflow management," Proceedings of the 6th International Conference, CAiSE '94. Utrecht, Netherlands, 6-10 June 1994.

[18] M. Hammer and J. Champy, "Reengineering the Corporation," Harper Business Publishers, 1993.

[19] E. N. Hanson, "Rule Condition Testing and Action Execution in Ariel," Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, June 1992.

[20] L. S. Homem de Mello, and A. C. Sanderson, "Representations for mechanical assembly sequences", IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, April 1991.

[21] M. Hsu, "Transaction Processing and Workflow Management," Proc. Workflow 94, Boston, March 1994. Proc. IEEE Compcon, 1991.

[22] H. Korth and A. Silberschatz, Database Systems Concepts, McGraw Hill (second edition), 1991.

[23] A. Kumar, "A New Approach for Conflict Resolution and Rule Processing in a Knowledge-Based System," Proceedings of the International Conference on Information Systems, Copenhagen, December 1990.

[24] L. O. Levine and S. S. Aurand,"Evaluating automated work-flow systems for administrative processes," Interfaces, vol.24, no.5:141-51, 1994.

[25] F. Leymann and W. Altenhuber,"Managing business processes as an information resource," IBM Systems Journal, Vol. 33, No. 2, 1994.

[26] F. Leymann and D. Roller,"Business process management with Flowmark," Proc. IEEE Compcon, March 1994.

[27] D. McCarthy and U. Dayal, "The architecture of an active, object-oriented database system," Proc. ACM SIGMOD, Portland, OR, 1989.

[28] R. Medina-Mora et al., "The action workflow approach to workflow management technology," CSCW 92 Proceedings, November 1992.

[29] Miranker, D., "TREAT: A better match algorithm for production systems," Proc. of AAAI 87 Conference on Artificial Intelligence, August 87.

[30] Miranker, D. and D. Brant, "An algorithmic basis for integrating production and database systems," Proc. Sixth International Conference on Data Engineering, February 1990.

[31] T. Murata, "Petri Nets: properties, analysis and applications", Proceedings of the IEEE, 77(4):541-580, April 1989.

[32] K. M. Olender and L. J. Osterweil, "Cecil: a sequencing constraint language for automatic static analysis generation", IEEE Transactions on Software Engineering, Vol. 16, No. 3, March 1990.

[33] V. N. Rajan and S. Y. Nof, "Minimal precedence constraints for integrated assembly and execution planning", IEEE Transactions on Robotics and Automation, Vol. 12, No. 2, April 1996.

[34] M. Rusinkiewicz, and A. Sheth, "Specification and execution of transactional workflows," Bellcore technical report, 1993.

[35] M. Schlatter, et. al., "The Business object management system," IBM Systems Journal, Vol. 33, No. 2, 1994.

[36] Arie Segev and J. Leon Zhao, "Rule management in expert database systems", Management Science, Vol. 40, No. 6, pp. 685-707, June 1994.

[37] T. Sellis, C.-C. Lin, and L. Raschid, "Coupling production systems and database systems: A homogeneous approach, IEEE Transactions on Knowledge and Data Engineering, 5(2), April, 1993.

[38] Sheth, Amit. Editor. Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems: State of the Art and Future Directions. Athens, GA, USA, 8-10 May 1996.

[39] Edward A. Stohr and J. Leon Zhao, "A technology adaptation model for business process automation", Proceedings of the 30th Annual Hawaii International Conference on Systems Sciences, January, 1997.

[40] D. Strong, "Decision support for exception handling and quality control in office operations," Decision Support Systems, 9 (1992) 217-227.

[41] W.M.P. van der Aalst, "Petri-net-based workflow management software", NSF Workshop on Workflow and Process Automation, May 8-10, 1996, Athens, Georgia.

[42] B. Vandenbosch and M.J. Ginzberg, "Lotus Notes and collaboration: le plus ca change," Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences, vol.3, pp. 61-71, Wailea, USA, 3-6 January, 1996.

[43] WFMC, "Workflow reference model", Technical report, Workflow Management Coalition, Brussels, 1994.

[44] M. Zisman, "Office automation: revolution or evolution," Sloan Management Review, 19, 3 (Spring 78), 1-16.