# Developing A Document-based Workflow Modeling Support System: A Case-based Reasoning Approach

**Jaeho Kim*, Woojong Suh**, Heeseok Lee***

**Graduate School of Management, KAIST, jhkim@kgsm.kaist.ac.kr, arizona@unitel.co.kr*

***POSCO Research Institute, wjsuh@mail.posri.re.kr*

**ABSTRACT**

A workflow model is useful for business process analysis and has often been implemented for office automation through information technology. Accordingly, the results of workflow modeling need to be systematically managed as information assets. In order to manage the modeling process effectively, it is necessary to enhance the efficiency of their reuse. Therefore, this paper creates a Document-based Workflow Modeling Support System (DWMSS) using a case-based reasoning (CBR) approach. It proposes a system architecture, and the corresponding modeling process is developed. Furthermore, a repository, which consists of a case base and vocabulary base, is built. A case study is illustrated to demonstrate the usefulness of this system.

## 1. INTRODUCTION

Workflow technology has gained its popularity due to the increased interest in business process reengineering and improvement of related technologies such as inter-networking and object-oriented technology [Jablonski and Bussler, 1996]. Many works agree that workflow models are useful for business process analysis and the origin of workflow technology is in office automation [Bracchi and Pernici, 1984; Ellis and Nutt, 1980]. Workflow technologies have been expected to facilitate enterprises' requests for competition by reducing the cost of doing business and by rapidly developing new services and products [Hales et al., 1997]. The technologies have been used as a practical means for achieving such requests primarily in the industrial sector rather than in the research domain.

Workflow modeling is a design activity, which belongs to an experience-rich domain. This means that it is learned by experience, like data modeling and object modeling. While experts in workflow modeling have been trying to develop a conceptual framework for workflow modeling for many years, the practice of workflow modeling is still somewhat *ad hoc*.

Case-Based Reasoning(CBR) systems [Kolodner, 1993; Korczak, 1989; Riesbeck and Schank, 1989] is an artificial intelligence technology that solves problems by using knowledge gained from solving similar problems in the past. Major activities of such systems include retrieving and selecting similar previous cases, adapting them to solve new problems. Therefore, it is a very useful tool for domains where expertise and experience are valuable and hard to acquire[Brown and Gupta, 1994]. In this respect, CBR is preferred over any other artificial intelligence approach, such as rule-based reasoning, for design problems.

The case-based approach for information system modeling automation has been found for data modeling [Paek, 1996; Han, 1997] and for object modeling [Han, 1998]. However, no research has been done for workflow modeling so far, even though it is regarded as very useful tool. Therefore, in this paper, we suggest a *Document-based Workflow Modeling Support System* (DWMSS) using the CBR approach.

The corporate documents are produced according to certain organizational processes [Uijlenbroek and Sol, 1997]. Furthermore, most formal business tasks are based on, or driven by, document flows [Sprague, 1995]. Thus, documents and business processes should be considered simultaneously for the analysis of a corporate information system [Sprague, 1995; Frank, 1997; Morschheuser et al., 1996]. For these reasons, a document-based workflow model can be effective in analyzing business requirements for developing information system. Therefore, in the next section, we introduce a document-based workflow model [Lee and Suh, 2001]. Section 3 describes the system architecture and modeling process of DWMSS. In Section 4, the

repository for this system is described: the structure of the case base and the case representation are in Section 4.1 and the vocabulary base is in Section 4.2,. In Section 5, we go into the details of the modeling process of DWMSS. Section 6 concludes the paper and points out some important issues for future research.

## 2. A DOCUMENT-BASED WORKFLOW MODEL

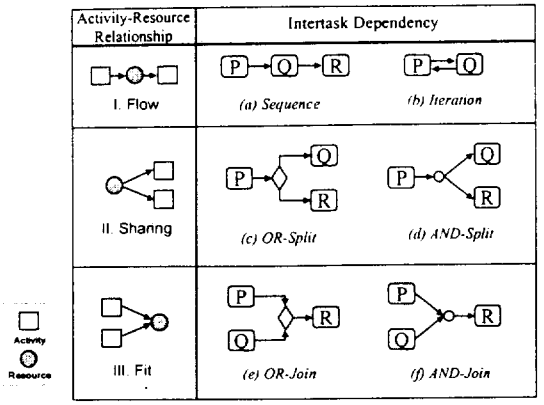
A document-based workflow model includes key elements of process models (based on [Armitage and Kellner, 1994; Curtis and Over, 1992], as shown in Table 1. But this model does not include the more detailed notations for task flow, such as intertask dependencies and simultaneity constraints. First, the intertask dependency is an important concept in showing how work is routed among tasks. It shows how one task performed diverges into several other tasks that follow. It also shows how several tasks converge into one task that follows. Second, when divergence or convergence takes place, the simultaneity constraints in intertask dependency show whether or not those flows are occur simultaneously. To do this, we extended Suh's model by adding the notations for intertask dependencies and simultaneity constraints.

<Table 1> Elements of Workflow Model

| Model Elements | Description |
|---|---|
| Task | A unit consisting of a workflow (operations or descriptions for human actions with documents. ) |
| Agent | An individual identified by hierarchical status in an organization. |
| Document | A view of information set which is determined in order to support a task. |

### 2.1 Intertask Dependency

Documents can go off on different routes and then merge back into a single route at a "rendezvous" point. In addition, a document can be split into multiple parts and merged back into a single part as it moves down the workflow river. This is done using splits and joints. In Figure 1, six types of intertask dependencies are shown.




<Figure 1> Intertask dependency types and grouping of them in terms of activity- resource relationship

Furthermore, we can group six intertask dependencies into three activity-resource (that is, task-document) relationships [Malone et al, 1997], as shown Figure 1. First, Sequence and Iteration intertask dependencies can be grouped together as a Flow activity-resource relationship. Second, OR-Split and AND-Split can be grouped together as a Sharing activity-resource relationship. Finally, OR-Join and AND-Join can be grouped together as a Fit activity-resource relationship.

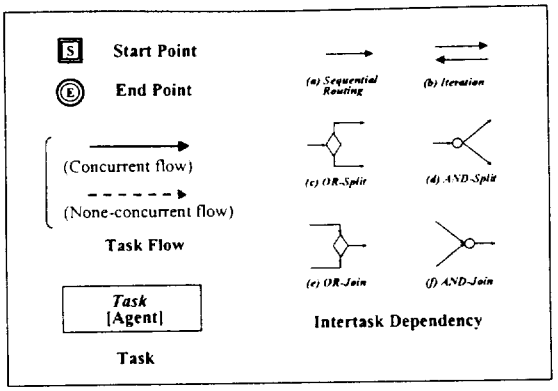
### 2.2 Simultaneity Constraints of Task Flow

Simultaneity constraint means that two tasks, Q and R, originating from task P, should be performed at the same time or that two tasks, P and Q, moving toward a task, R, should be performed at the same time in order for task R to be performed properly.

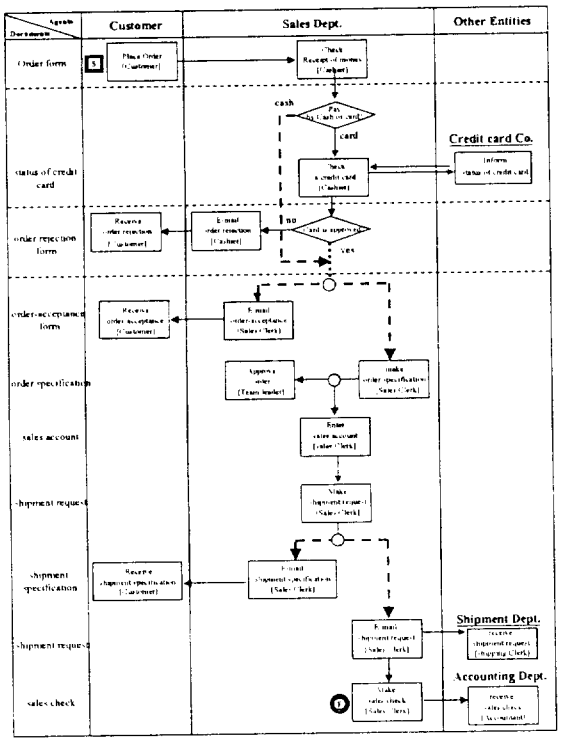
As we see in Figure 1, AND-Split shows that the completion of task P enables the execution of two task, Q and R, and OR-Join shows that the completion of two task, P or Q, enables the execution of task R. On the basis of simultaneity constraints, we can classify both of these dependencies into two categories again: Concurrent flow and Non-concurrent flow.

### 2.3 Extended Notations

Lets examine the extended notations of a document-based workflow model in Figure 2, where we can see that the notations for the six types of the intertask dependency and two types of simultaneity constraints are added. In Figure 3. we demonstrate how a real-world case, '*Order Processing*' workflow in the Internet Business of K Bookstore Co. Ltd., can be modeled with these extended notations.




<Figure 2> Extended notations for Model Element



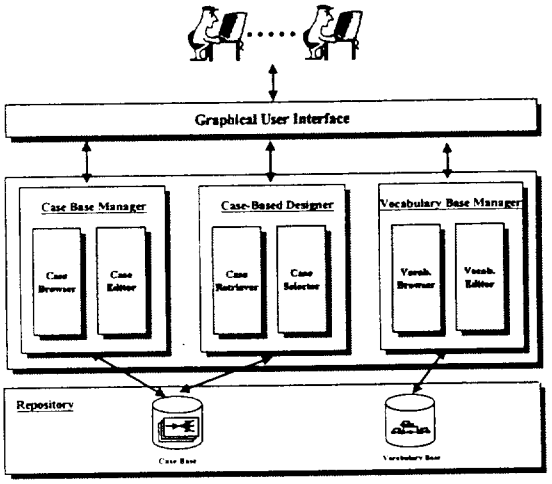

<Figure 3> Workflow Model for '*Order Processing*' in the Internet Business of K Bookstore Co. Ltd.

## 3. A DOCUMENT-BASED WORKFLOW MODELING AND SUPPORT SYSTEM

### 3.1 System Architecture

DWMSS consists of five major modules and two kinds of repositories, as shown in Figure 4.

The *Graphical User Interface* module allows the designer to access each module by providing interactive question-answering functions. It receives the users' requirements for workflow modeling, asks the user for specific information, accepts the answer, presents the workflow model description, and shows the designed workflow model.




<Figure 4> System Architecture of DWMSS

The *Case Base Manager* is made up of two modules: the Case Browser and the Case Editor. The *Case Browser* module provides facilities for browsing the cases in the case base. The *Case Editor* module provides facilities for deleting, editing workflows stored in the case base, and adding new cases into the case base.

The *Case-Based Designer* is made up of two modules: the Case Retriever and the Case Selector. The *Case Retriever* module is to retrieve similar cases given a set of requirements of the current problem by calculating the similarity score and to maintain a list of similar cases. The *Case Selector* module ranks the retrieved cases and presents the ranking to the user.

The *Vocabulary base manager* is made up of two modules: the Vocabulary Browser and the Vocabulary Editor. The *Vocabulary Browser* module provides facilities for browsing the vocabulary in the Vocabulary base, such as task names (e.g., 'receive order', 'check credit', etc.) and the nouns/verbs which constitute the task names. The *Vocabulary Editor* module provides facilities for deleting, and editing vocabulary stored in the vocabulary base and adding new vocabulary into the vocabulary base.

The *Repository* is made up of two bases: the Case Base and the Vocabulary Base. The *Case Base* stores a set of cases and is organized in a hierarchical tree format, layers of which are arranged in order of the abstraction level of the business domain, such as the business area, functional area, function, workflow, and task. This structure resembles a conceptual *is-a* hierarchy. The *Vocabulary base* maintains the task

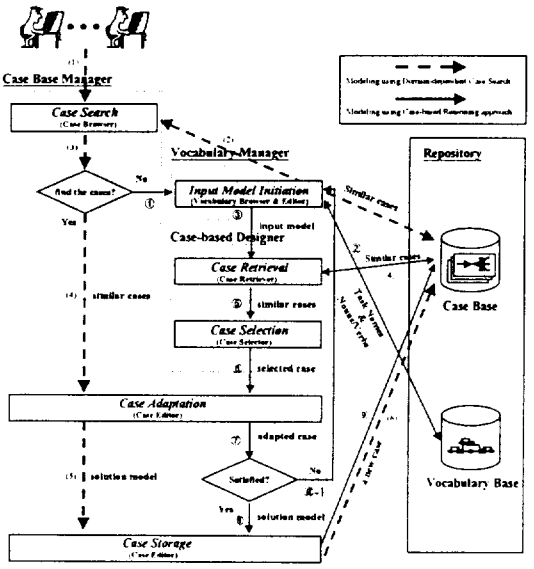
names and nouns/verbs that constitute the task names. Here each noun and verb is related to its similar terms, e.g., the noun 'inventory' is related to 'stock, stockpile, etc.', and the verb 'check' is related to 'look_into, confirm, etc.'.

### 3.2 Modeling Process

A workflow modeling process of DWMSS is illustrated in Figure 5. DWMSS employs two kinds of modeling strategies; modeling using a domain-dependant case search and modeling using a case-based reasoning approach. Normally the user starts modeling using the domain-dependant case search, because it is a faster and easier way to model and it is more familiar and understandable to the user, as long as similar cases can be found. If users cannot find similar cases using the domain-dependant case search, then the user should rely on the second strategy; modeling using the case-based reasoning approach. Next we will give an overview of the modeling process of DWMSS and the details will be described later.

First, we start modeling using the domain-dependent case search, where the user scans the case base using the case browser. Even if the user could find a similar case, generally it does not exactly fit the user's requirements. But it is not easy for the users to guess what portions of the retrieved case should be changed, supplemented and deleted. Therefore, the user needs to use the case adaptation process of modeling using the case-based reasoning approach. Once the user adapts a retrieved case, it should be stored in the case base for future reuse.

Second, if the user cannot find a similar case in the case base, then the user should depend on modeling using the case-based reasoning approach. First, the user should initiate the input model that will be used as input data for case retrieval. To initiate an input model, first the user guesses what tasks should be included in the solution model and then, search for the names of those tasks in the vocabulary base using a vocabulary browser. With these tasks completed, the user initiates the input model by interconnecting them in logical order in the view of intertask dependency. Next, the user inputs this input model into the CBR system, and then the CBR system retrieves similar cases from the case base according to the case retrieval rules, and lists them in the order of the similarity score and selects the best case. If the retrieved cases are unsatisfactory for the user's requirement, they should be adapted to the user's requirements. But if the adapted case is still unsatisfactory, the user goes back to the input model initiation process to initiate an improved input model referring to the retrieved cases. Finally, if the solution model is obtained, it should be stored in the case base for future reuse.




<Figure 5> Modeling Process of DWMSS

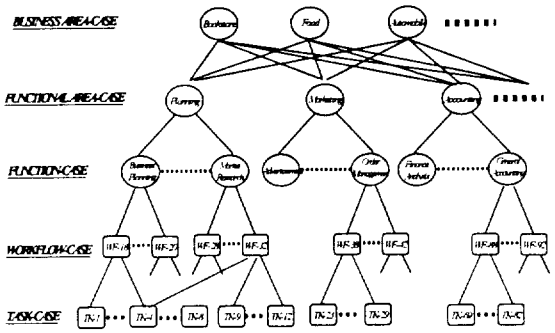
## 4. A Repository

### 4.1 Case Base

In DWMSS, the case base is well organized in the form of a hierarchical case tree from the top layer (Business Area cases) to the bottom layer (Task cases), and resembles a conceptual *is-a* hierarchy, called the domain-dependent case hierarchy, as shown in Figure 6.

The upper three layers (Business Area, Functional Area and Function layer) represent the more abstract, more general cases and are called the *domain cases*, the lower two layers (Workflow case and Task case) represent the less abstract, more specific cases. In the view of the bottom-up case construction, a workflow case in common several task cases, and a task case could be used in several workflows, a function case consists of several workflow and a functional area case consists of several function cases and so on. If a new workflow case is created, it is saved in the relevant location in the case base by routing it along the hierarchical path from the top layer, Business Area case, to the workflow layer. Each domain case has its identifier, domain features, and a brief description of the domain it represents.

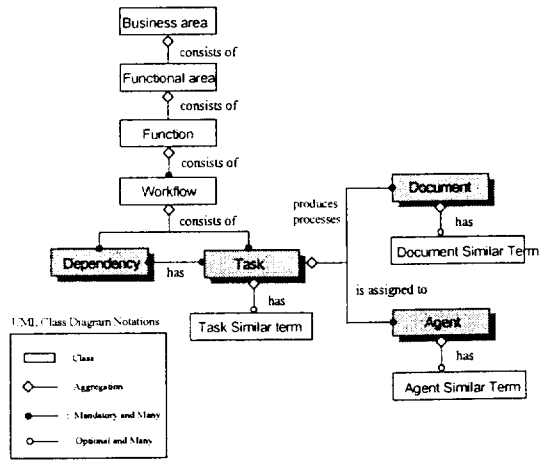
Below the domain case hierarchy, there is the workflow and task case hierarchy. Workflow features, which describe a workflow, include the function it belongs to, name, description, head task, any other workflows that are connected to this workflow, and the utilization degree that represents the number of references to this case.



<Figure 6> The Structure of the Case Base

In this paper, the information of each case is represented using a frame. For this, we present the Workflow Meta-model. In this paper, we represent the workflow meta-model using notations for Calss Diagram of UML (Unified Modeling Language) [Booch et al., 1997], as shown in Figure 7. The shadowed classes, such as task, dependency, document and agent, are those that are employed as indices for case retrieval.

As we see in figure 7, a workflow consists of tasks, intertask dependencies, agents and documents. Each task has its similar term set, each member of which has the term similarity weight (a range of 0-1) according to how analogous it is to that task name in terms of semantics. And these weights are determined by administrator's experience.




<Figure 7> Workflow Meta-Model

Each task has the documents that are the input and output artifacts to be processed or produced by task itself. Each task is performed by an agent or agents who may be a person or a group. And each document and agent also has its similar term set of names, as in tasks. These similar term sets are the principal indexes for retrieving similar cases from the case base using the CBR approach.

### 4.2 Vocabulary Base

To understand what the vocabulary base is and how it can be used, we need to understand its structure as shown in Figure 8 which shows the conceptual data model given in ER (entity-relationship) notation. The ER schema has been extended to represent fuzzy relationships (denoted with a plus).

The *Case* entity represents the workflows stored in the case base. Each *Case* is located by its primary key *CId* (case ID; that is, the workflow ID). This internal code reference is completely transparent to the user, who refers to workflows by their names. The *is_A* Boolean relationship maps specialization and inheritance among workflows, as a recursive and many-to-many relationship, to capture multiple derivation.

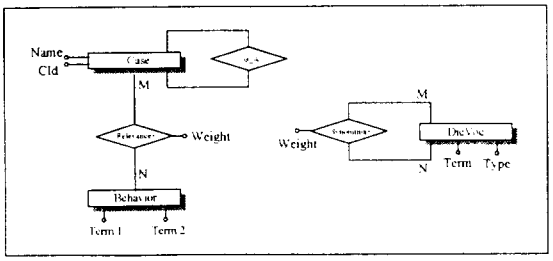
The *Behavior* entity represents an action describing the service that the task provides. Its name is composed of a term pair (term 1 and term 2), which is constructed as verb/noun pairs and correspond to task names. For example, it describes actions like *'take order'* and *'make shipment_ request'* and so on. Also each *Behavior* has its synonymous terms, which means they do the same kind of job processing. For example, the *Behavior* '*receive order*' has its synonymous terms '*get order*', '*obtain order*' '*take purchase_order*' and so on.

The *Dicvoc* entity represents terms of verbs and nouns in the vocabulary base. Each verb and noun can have its synonyms. For example, while the verb '*receive*' has its synonyms '*get, have, obtain*', the noun '*order*' has its synonyms '*purchase_order*'. Furthermore, each verb (or noun) is related several nouns(verbs) and each combination of a verb(noun) and one of those nouns(verbs) may form a behavior. For example, while the verb 'receive' is related to several nouns 'order, receipt, cash, .....' in the form of task names, the noun 'order' is related to several verbs 'receive, reject, transfer, .....' as well. Therefore, when the user thinks up a verb (or a noun) which constitutes a *behavior* name (that is, a task name), the user can easily find the counterpart noun (or verb) and form a *Behavior*

name by combining both of them.

The extended relationship *Relevance+* partitions the *Behaviors*, or tasks, in the context of each workflow's goal. This extended relationship captures the concept of *weight*, which is interpreted as the relevance assigned to *Behavior* in a *Case* (that is, workflow), or the membership degree of a *Behavior* to *Case*. In other words, in a workflow, the roles of some tasks are more decisive than the other tasks in order to achieve the goal of that workflow. For example, as we see in the '*order_processing*' workflow of Figure 3, the task *'make order_ specification'* is a more important factor than the task *'check production plan'* in that, without the formers, the whole processing cannot be done completely. Therefore, the stronger the relevance of a task (task weight) in a particular workflow, the greater the membership value for that workflow. The *weight* belongs to the unit interval.

The extended relationship *synonymia+* represents the synonyms among task names or among verbs and nouns, respectively. This is a binary, recursive relationship. The *synonymia* between two terms is measured by a *weight* belonging to the unit interval. Again, we interpret the imprecision expressed by this weight in terms of fuzziness. In other words, the synonymia degree, the *weight* of the relationship itself, is thought of as the membership value of a synonym for a term to the term itself. *Synonymia* is analogous to a crisp equivalence relationship. Thus, for each term, it is possible to determine a similar term set; that is, a fuzzy set representing all synonyms for that term.

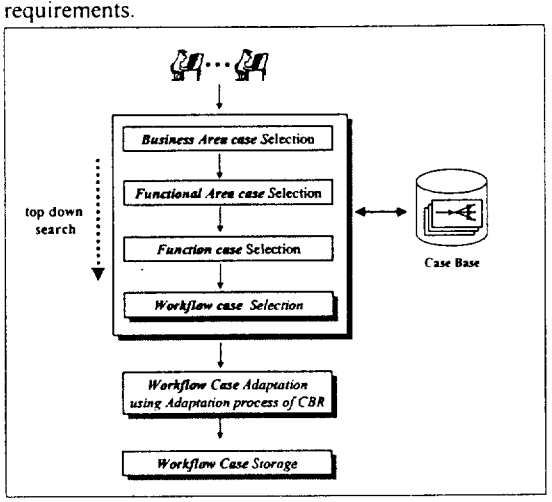



<Figure 8> ER extended conceptual model for Vocabulary base

## 5. WORKFLOW MODELING PROCESS

### 5.1 Modeling using a Domain-dependent Case Search

Modeling using the domain-dependent case search is a kind of breadth-first search, which examines all of the nodes (cases) in a case tree (the case base) beginning with the root node (Business layer), as shown in Figure 9. If the user can find a similar case, he can reuse it without any change or adapt it according to his requirements.




<Figure 9> Domain-dependent case search process

Because, in general, the retrieved case does not exactly match the user's requirements, adaptation rules are needed to change the differences, fill in the missing parts and remove the unnecessary parts. In modeling using the domain-dependent case search, the retrieved case will be adapted by using the adaptation process of CBR, through which the unmatched parts could be substituted with a reusable part of any other case.

Finally, if the solution model is created through the above processes, this new case should be saved in the case base for future reuse.

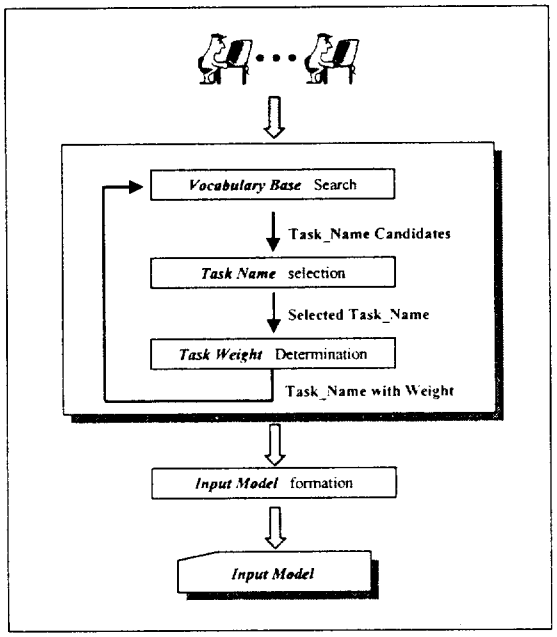
### 5.2. Modeling using Case-based Reasoning Approach

When the user cannot find similar cases using the domain-dependent case search, the user should employ the case-based reasoning approach instead. This strategy may be regarded as a kind of the bottom-up approach in some ways, from the point of view that it starts from the tasks which constitutes the bottom layer of the case hierarchy in the case base.

#### 5.2.1 Input Model Initiation

As shown in Figure 10, to initiate an input model, the user should find the tasks that are supposed to be the tasks of the solution model. But it is not so easy for the user to guess the tasks just by using his imagination. The user needs a mechanism to help him with this job. For this, the vocabulary base, the repository including the task names, is structured and managed.

First, the user guesses what tasks the solution model would include in it and finds each of those tasks' names with the help of the Vocabulary base manager of the system. If the user inputs a verb (or noun), the system suggests the counterpart nouns (or verbs) which are connected to that verb (or noun). For example, if the user inputs the verb 'take', the system will show the related nouns, "receipt" from "take receipt' and "report" from "take report" and so on.




<Figure 10> Input Model Initiation process

Next, the user should determine the weight of the selected task in terms of how that task is important to achieve the goal of the target workflow. For example, in '*order_processing*' workflow, the task '*make order_specification*' is a more important factor than the task '*check production plan*' in that, without the former, the whole processing can not be done completely.

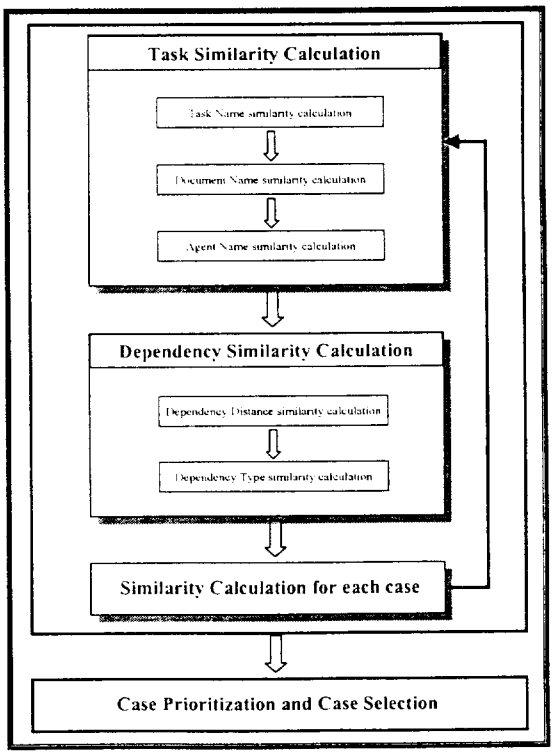
The user repeats the above process to get several tasks that are needed to formulate an input model. When the user thinks that the necessary tasks are all ready, the user begins to initiate an input model by setting up the intertask dependencies among tasks in terms of how those tasks should be interconnected for the job to be done.

### 5.2.2 Case Retrieval and Case Selection

As shown in Figure 11, by inputting an input model, the system calculates the similarity score for every case in the case base using the case retrieval rules which are in the form of a formula and then ranks the retrieved cases in the order of the similarity score. Among them, the user selects the case with the highest similarity score as the most similar case.

#### 5.2.2.1 Task Similarity Calculation

As shown in Figure 11, the task similarity calculation consists of three steps: task name, document name and agent name similarity calculation. The task name similarity score is calculated by comparing the name of the specific task $i$ in the input model with the task names in the *similar term set* (as shown in Formula 1 and 2) of a task in a workflow case. If the name of the specific task $i$ in the input model is included in this similar term set, it can get a correspondent similarity score. For example, a task name 'receive order' has its similar term set of {receive order, 1, take order, 1, request order, 0.8, .....}, where the similar term 'take order' has the similarity weight 1 and 'request order' has 0.8, and so on.




<Figure 11> Case Retrieval and Case Selection Procedure

The document and agent similarity score calculation process is based on the same concept as the task name similarity score calculation process, as shown in Formula 3 and 4.

$$T_i = Tn_i + D_i + A_i$$

$$D_i = \{(\sum_{a=1}^{m} IDn_{ia})/m + (\sum_{b=1}^{n} ODn_{ib})/n\}/2$$

$$A_i = \sum_{c=1}^{l} (An_{ic})/l$$

$T_i$ : similarity degree of Task i
$Tn_i$ : name similarity degree of Task i
$D_i$ : name similarity degree of Document of Task i
$R_i$ : name similarity degree of Agent of Task i
$IDn_{ia}$ : name similarity degree of Input Document
$ODn_{ib}$ : name similarity degree of Output Document
$An_{ic}$ : name similarity degree of Agent
i : index of task
a : index of Input Document
b : index of Output Document
c : index of Agent
m : number of Input Documents processed by Task i
n : number of Output Documents produced by Task i
l : number of Agent of Task I

<Formula 1> Task Similarity Calculation Formula

Similar_Term (Tn)={$Tn_1$,$w_1$,$Tn_2$,$w_2$,$Tn_3$,$w_3$,.....$Tn_k$, $w_k$}

Tn : task name
$Tn_k$ : the $K_{th}$ member of similar term set
$w_k$ : weight of the $k_{th}$ member of similar term set

(Example)
Similar_Term(take order)={take order, 1.0, receive order, 0.8, get order, 0.6, have order, 0.5,.....}

<Formula 2> Similar Term Set Representation and Example

$$D_i = \{(\sum_{a=1}^{k} IDn_{ia})/K + (\sum_{b=1}^{l} ODn_{ib})/l\}/2$$

$D_i$ : name similarity degree of documents of Task i
$IDn_{ia}$ : name similarity degree of Input Document ia
$ODn_{ib}$ : name similarity degree of Output Document ib
i : index of task
a : index of Input Document
b : index of Output Document
k : number of Input Documents processed by Task i
l : number of Output Documents produced by Task i

<Formula 3> Document Similarity Calculation Formula

$$A_i = \sum_{c=1}^{l} (An_{ic})/l$$

$A_i$ : name similarity degree of Agents of Task i
$An_{ic}$ : name similarity degree of Agent ic
i : index of Task
c : index of Agent
l : number of Agents of Task i

<Formula 4> Agent Similarity Calculation Formula

**5.2.2.2 Intertask Dependency Similarity Calculation**

The intertask dependency similarity score is calculated by comparing the distance and type of the intertask dependency between two tasks in the input model with those of the intertask dependency between the corresponding two tasks in a workflow case, as shown in Formula 5. In order to get the intertask dependency similarity degree, therefore, there should be the correspondent two tasks in a workflow case.

The distance similarity degree of a specific intertask dependency is decided by how similar the distance of a specific intertask dependency in the input model is with the correspondent one in a workflow case. The intertask dependency distance of the specific two tasks is calculated by the number of intertask dependencies between them and if there are many routes between two tasks, the shortest one is chosen as a correspondent comparable intertask dependency.

For example, in Figure 12, There are two routes from task A to B, [A =>C'=>D'=>B] and [A =>C'=>B]. But the comparable intertask dependency would be [A =>C'=>B], because its route is shorter than the other. Therefore, the distance of the intertask dependency between task A and B is 2 and its similarity score is 0.8, as shown in Figure 13.

The similarity score of the intertask dependency type is calculated by applying the matrix for the similarity score of the intertask dependency type, as shown in Figure 14. As we have seen before in Figure 1, the intertask dependency type could be categorized in terms of the activity-resource relationship among tasks. If both intertask dependencies are in a same category, the similarity score will be 1 or 0.5.

$$P_i = Pd_i + \{(L\_Pt_i + L\_Pc_i)/2 + (R\_Pt_i + R\_Pc_i)/2\}$$

$P_j$ : dependency similarity degree of intertask dependency j
$Pd_j$ : dependency distance similarity degree of intertask

dependency j

$L\_Pt_j$ : dependency type similarity degree of the left-side of intertask dependency j

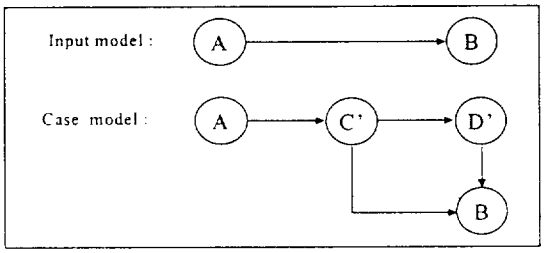
$L\_Pc_j$ : dependency cardinality similarity degree of the left-side of intertask dependency j

$R\_Pt_j$ : dependency type similarity degree of the right-side of intertask dependency j

$R\_Pc_j$ : dependency cardinality similarity degree of the right-side of intertask dependency j

j: index of Intertask Dependency

<Formula 5> Intertask Dependency Similarity Calculation Formula




<Figure 12> Example of the Intertask Dependency Distance

| Dependency Distance | 1 | 2 | 3 | Over 4 |
|---|---|---|---|---|
| Similarity Score | 1 | 0.8 | 0.6 | 0 |

<Figure 13> Matrix for Similarity Score of the Intertask Dependency Distance

| Activity-Resource Relationship | | FLOW | | SHARING | | FIT | |
|---|---|---|---|---|---|---|---|
| A \ B | | SE | IT | OS | AS | OJ | AJ |
| FLOW | SE | 1 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 |
| | IT | 0.5 | 1 | 0.0 | 0.0 | 0.0 | 0.0 |
| SHARING | OS | 0.0 | 0.0 | 1 | 0.5 | 0.0 | 0.0 |
| | AS | 0.0 | 0.0 | 0.5 | 1 | 0.0 | 0.0 |
| FIT | OJ | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 0.5 |
| | AJ | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 | 1 |

<Figure 14> Matrix for Similarity Score of the Intertask Dependency Type

#### 5.2.2.3 Total Similarity Calculation

As shown in Formula 6, the total similarity score of a case model is calculated by combining the task similarity score which is weighted in terms of each task's importance in a workflow case and the intertask dependency similarity score.

Once we find the similarity in some tasks, we can judge that two workflow models are similar, because the similarity between intertask dependencies is dependent on the similarity in tasks. Therefore, the similarity between intertask dependencies is not a decisive factor, just an additional factor which increases the total similarity degree. The more a workflow case includes tasks with higher weight (that is, key tasks), the higher the total similarity score it gets.

$$TSS = \sum_{i=t}^{m} T_i \bullet W_i + \sum_{j=t}^{n} P_j \quad (i=1,2,\ldots,m\ j=1,2,\ldots n)$$

TSS : total similarity score

$T_i$ : similarity degree of Task i

$W_i$ : weight of Task i

$P_j$ : similarity degree of Intertask Dependency j

m : the number of Task

n : the number of Intertask Dependency

<Formula 6> Total Similarity Calculation Formula for Each Case

#### 5.2.3 Case Adaptation

The selected case should be modified by combining the input model and the selected model according to adaptation rules. The principles for combination of the input model and the selected model are as follows:

First, those that are determined only by the user, (e.g. task name, document name, role name) are combined based predominantly on the input model.

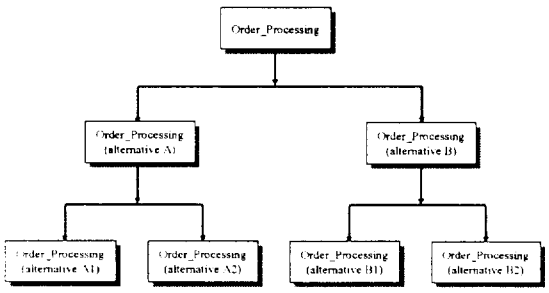
Second, those that are determined by the business policy rather than the user's judgement (e.g. intertask dependency type and cardinality) are combined based predominantly on the input model.

Third, for those, which are thought of as 'the more, the better' cases (e.g. the similar term set of task, document and agent name), the input model and the retrieved model are combined on a par with each other.

In general, most of the combined models are not so satisfactory for the user's requirements. So the user should find out the unsatisfactory parts and adapt the combined model. For this, the user uses again the CBR approach to get alternatives for unsatisfactory parts from the other cases in the case base. If the user finds alternative parts, the user can adapt the combined model using them. By repeating this process, the user could proceed to a more complete solution model, step by step.

### 5.2.4 Case Storage

The storage rule is needed to add a new case, or solution model, to the case base and to manage all workflow models systematically for future reuse. We can more systematically represent and classify the workflow models by using *specialization and inheritance* mechanisms, as shown in Figure 15, which means that, like in object-oriented methodology, a workfow model could be created differently to ensure versatile workflow models.




<Figure 15> Specialization of workflow

## 6. CONCLUDING REMARKS

In this paper, we proposed the workflow model reuse methodology using a case-base reasoning approach. With the help of DWMSS, enterprises and consulting companies can reduce the cost and time needed to design a workflow model from a draft. Furthermore, these past workflows are very useful materials for employee training, helping them understand the whole picture of their job easily. And these accumulated workflows are valuable information assets of a company, thus they should be well organized and maintained.

The following are some further research areas on the basis of this paper. First, the workflow model case representation should be reinforced to cover as many workflow models as possible. Second, the case retrieval method needs to be extended to cover the overall structure of workflow, which means that the whole realm of the key tasks and their intertask dependencies should be considered in calculating their similarity scores. Third, DWMSS needs to be more automated to be more helpful to users because this system still needs more interaction between humans and the system.

## REFERENCES


Armitage, J. and M. Kellner, "A Conceptual Schema for Process Definitions and Models", *In Proc. 3rd Int'l. Conf.. on the Software Process*, Reston, Virginia, USA, October 1994, pp.153-165.

Booch, G., I. Jacobson and J. Rumbaugh, *UML version 1.0*, Rational Software, 1997.

Bracchi, G. and B. Pernici, "The Design Requirements of Office Systems," *ACM Transactions on Office Information Systems*, vol.2, no.2, 1984, pp.151-170.

Brown, C.E. and U.G. Gupta, "Applying Case-Based Reasoning to the Accounting Domain", *Intelligent Systems in Accounting, Finance and Management*, vol.3, 1994, pp.205-221.

Curtis, B., M. I. Kellner, and J. Over, "Process Modeling", *Communications of the ACM*, Sept. 1992, vol. 35, No.9, pp.77.

Ellis, C. and G. Nutt, "Office Information Systems and Computer Science", *ACM Computing Survey*, vol.12, no.1, 1980, pp.27-60.

Frank, U., "Enhancing Object-Oriented Modeling with Concepts to Integrate Electronic Documents", *In Proc. of the 30th Hawaii Inte'l Conf. on System Science*, vol.6, 1997, pp.127-136.

Lawrence, P., *Workflow Handbook 1997*, John Wiley & Sons Inc., pp.27-32.

Han, J., "Case-Based Reasoning Framework for Data Model Reuse", In Proc. of Korea Expert System Society, Dec. 1997, vol.3, no.2, pp.33-55.

Han, J., *Object Model Reuse by Case-Based Reasoning*, Dissertation, Ajou University of South Korea, 1998, pp.48-82.

Lee, H. and W. Suh, "A Workflow-Based Methodology for Developing Hypermedia Systems", Journal of Organizational Computing and Electronic Commerce, vol.11, no.2, 2001, pp.77-105.

Jablonski, S. and C. Bussler, *"Workflow Management: Modeling Concepts, Architecture and Implementation"*, International Thomson computer press, London, 1996, pp.3-12.

Kolodner, J., *Case-Based Reasoning*, Morgan Kaufmann, San Mateo, CA., 1993. pp.3-30.

Korczak, J.J., L.A. Maciaszek, and G.L. Stafford, "Knowledge Base for Database Design", *In Proc. of Int'l Symposium on Database Systems for Advanced Applications*, 1989, pp.61-68.

Malone, T. W. and K. Crowston, and J. Lee et al., "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes", MIT Center for Coordination Science, Jan. 1997, (http://ideas.uqam.ca/ideas/data/Papers/wopmitccs198. html).

Morshheuser, S., H. Raufer, and C. Wargitsch, "Challenges and Solutions of Document and Workflow Management in a Manufacturing Enterprise: A Case Study," *In Proc. 29th Hawaii Int'l. Conf. on System Sciences – Digital Documents*, 1996, pp.4-12.

Paek Y., J. Seo, and G. Kim, "An expert system with case-based reasoning for database schema design", *Decision Support Systems*, vol.18, 1996, pp.83-95.

Riesbeck, C.D. and R. Schank., *"Inside Case-Based Reasoning"*, Hillsdale, NJ: Erlbaum, 1989, pp.25-26.

Sprague R.H., "Electronic Document Management: Challenges and Opportunities for Information Systems Managers", *MIS Quarterly*, Mar. 1995, pp.29-49.

Uijlenbroek, J.J.M. and H.G. Sol, "Document Based Process Improvement in the Public Sector: Settling for the Second Best is the Best You Can Do," *In Proc. of the 30th HICSS*, vol.6, 1997, pp.107-117.