워크플로우 관리 시스템에서의 실행 중 프로세스 정의 수용

(Run-Time Process Definition Accommodation in Workflow Management Systems)

한 동 수 † 심 재 용 ††
(Dongsoo Han) (Jaeyong Shim)

요 약 전통적인 워크플로우 패러다임에 있어서는 하나의 워크플로우가 자동으로 실행되기 위해서는 사전에 해당 워크플로우 템플릿이 정의되어야 한다. 하지만 많은 경우에 있어서 프로세스 생성시에 전체 워크플로우를 명확하게 정의할 수 없는 상황이 발생한다. 그러한 경우에는 워크플로우 실행 중에 부분적인 워크플로우가 정의되고 전체 워크플로우가 이들 부분적인 워크플로우 파편을 통합하여 유도되는 것이 적절하다. 워크플로우 관리시스템이 그러한 상황에 대처하기 위해서는 실행 중에 워크플로우를 정의하는 기능을 갖출 필요가 있다. 본 논문에서는 워크플로우 관리시스템에서 실행 중 워크플로우 정의 기능을 어떻게 수용할 것인지에 관해서 소개하고 그러한 기능을 수용하는 워크플로우 관리 시스템이 어떠한 구조이어야 하는지에 관해서 논의한다. 본 논문에서는 또한 실행 중 워크플로우 정의를 위한 세가지 워크플로우 파편 템플릿이 소개되며 이것들로부터 전체 워크플로우를 유도하는 알고리즘이 고안되었다. 실행 중 워크플로우 정의 기능을 구현하기 위해서는 커넥터 메커니즘이 고안되었으며 이것의 구조와 기능도 소개된다.

Abstract In conventional workflow paradigm, workflow template has to be defined at process build time before the workflow to be executed automatically. However there are many cases that the whole workflow cannot be defined at process build time clearly. Rather, it sometimes more proper that the parts of the workflow are defined at run time and the whole workflow is deduced from integrating the partly defined workflow fragments. For a workflow management system to manage the situation, it has to be equipped with run time process definition capability. In this paper we show how to accommodate run time process definition capability in workflow management system and what the system should be like to accommodate it. Three workflow fragment templates are suggested for run time process definition and an algorithm for deriving a workflow from the workflow fragments is illustrated. Connector facility is devised as a means to implement process definition at run time and the structure of the connector facility is illustrated with its functions.

1. Preface

Even though workflow concept has been widely spread nowadays, business process automation using Workflow Management System(WfMS) is not so common in large scale enterprises[1, 2, 7]. As a result, many business processes for large scale enterprises still remain in non-automated state. We may ascribe the reason to the weak intention of the top manager to support the automation of the business processes of his company, but we believe that the limited capability of the current workflow management systems is more serious hurdle for that.

We used to classify workflows as production workflow, administrative workflow and ad hoc workflow according to the properties of workflow and many workflow management system vendors

† 종신회원 : 한국정보통신대학원대학교 공학부 교수
dshan@icu.ac.kr
†† 비 회 원 : 한국정보통신대학원대학교 공학부
jacyong7@icu.ac.kr
논문접수 : 2000년 1월 26일
심사완료 : 2000년 9월 27일

insist that their system is better suited to a certain type of workflow[1, 2]. But usually, many workflows of large scale enterprises are in the mixed form of the workflow types and the ad hoc workflow is the major hurdle to automate them. Dynamic reconfiguration[11, 12] is often sited as a solution to cope with the situation. But we found that the dynamic reconfiguration is not enough and the conventional workflow paradigm in which workflow has to defined at build time[3, 4, 7, 10] whole at once for the automation does not fit to many cases. In some cases, process definition at run time is a normal behavior in real situation. Only a part of the process is defined by a workflow participant and the whole workflow could be derived by integrating the workflow fragments defined by workflow participants at run time. What is important in workflow management system is to provide a mechanism and functions to automate the workflows on computers connected in a network by connecting the workflow fragments defined at run time.

The fact that the whole workflow can be obtained by integrating workflow fragments defined at run time provides us totally different view on workflow paradigm. That is, BPR(Business Process Reengineering) in which traditional business process are redesigned to new processes to improve their efficiency is the not the preliminary step to automate a business process but could be the intermediary step to refine the captured current process. Thus the overhead of BPR to understand current process can be eliminated if the workflow can be derived automatically from the workflow fragments defined at run time where a business process should be automated even without predefined process at all.

In this paper we introduce the notion of Run-Time Process Definition(RTPD) and the functions that workflow management system has to provide for the support of the run time process definition. Using RTPD, workflow participants can define processes at run time as well as process build time. As a means to implement RTPD we propose connector facility in that three types of workflow fragments are suggested as a simplified way of workflow definition at run time and an algorithm to integrate the workflow fragments into a workflow process automatically has been devised. The way of process definition at run time is allowed only in restricted form because only workflow fragments are defined by general workflow participants and the fragments should be integrated to generate a workflow. The defined workflow fragments are registered and managed in connector facility to be used afterwards like workflow templates are handled in workflow system. Thus the connector facility can be regarded as a semi-workflow system including the role of the inbox of a department.

For the RTPD to be integrated with workflow management system, the workflow management system has to be extended appropriately to accommodate it. The structure of ICU/COWS(Connector Oriented Workflow System of Information and Communications University) is introduced and we show how we extend the system to accommodate the RTPD. We found the architecture of ICU/COWS in which workflow task managing instances is generated in the form of object instance and the whole task managing instances for a workflow instance is controlled by a global task managing instance is very flexible to implement RTPD. Note that, in RTPD the Task Managing Instance(TMI) should be generated at run time and the TMIs generated at run time should be treated in the same way as the TMIs generated from the workflow templates defined at workflow build time.

The paper is organized as follows. In section 2, we describe run time process definition and its benefits. In section 3, we explain the concept and usage of the connector facility in a workflow system. In section 4, we describe ICU/COWS and the extension of the system to accommodate the connector facility. In section 5, we show how an example workflow can be implemented in the connector-oriented workflow system. In section 6,

we describe related works which have been performed on workflow model induction. We discuss conclusion in section 7.

2. Run-Time Process Definition and its Benefits

We define RTPD as the action of defining new workflow processes or adding new workflow paths to the predefined workflow processes at run time of workflow instances. Almost all the definitions that can be defined at process build time can be defined also at run time. Thus the definition covers much wider range than the dynamic reconfiguration of workflow management system in which only partial modification to the predefined workflow templates or running instance is the main interest. Besides, in RTPD the registry of the defined processes should be mandatory because they have to be used by users and manipulated in automatic process derivation. Figure 1 compares the differences between dynamic reconfiguration and RTPD in several aspects.

Items	RTPD	Dynamic Reconfiguration
Definition Time	Run time	Run time
Predefined Workflow Template	Yes or No	Yes
Definition Scope	Broad	Narrow
Influences to other Workflow Instances	No	Yes
Management of Defined Processes	Mandatory	Optional

Fig. 1 Comparison matrix between RTPD and dynamic reconfiguration

We can expect several additional benefits when the RTPD facility is accommodated in workflow management system. First, many real world workflows of which automation is given up because of its property requiring RTPD can be automated. Second, current workflow paths can be derived automatically from the workflow fragments defined at run time if some deliberately devised facility is provided. The automatically derived workflow processes can be the start point of BPR. Third, workflow process definition overhead can be diminished drastically if workflow management system can derive workflow processes from the workflow fragments automatically. Fourth, more flexible integration of various workflow types such as production, administrative and ad hoc workflow is possible.

3. Connector Facility

A connector is defined as a storage and services either to store incoming work items or data from the other departments or to access the storage[5]. The connector facility plays a key role in implementing RTPD in ICU/COWS. The primary function of the connector is to connect inter-department workflow with the following functions. First, the connector has a storage service to store the work items to be handed over from several departments to one department. When it stores the work items, for the same workflow instances to be connected it keeps the workflow instance ID and propagates it until the process instance is completed. Second, the connector has to provide some means to access the data it stores. For that it provides APIs(Application Program Interface) for application programs or user interfaces for users. Third, the connector has functions to register events and keep statistical information on historical events. This information not only shows the current situation but also can be used as basic data for extracting the execution path to enable BPR. Fourth, the connector provides facility to define structured data easily. The defined structured data for each connector are manipulated and managed by the workflow system in integrated manner. Although the way of structured data managing is similar to that of workflow relevant data, it differs from the workflow relevant data in that it can be defined at run time. Fifth, the connector has a function to keep the workflow fragments defined at run time and provides some facilities to reuse the workflow fragments easily. Using this facility, the whole workflow can be built

from the defined workflow fragments in an incremental way. Figure 2 shows the layered view of the connector facility.

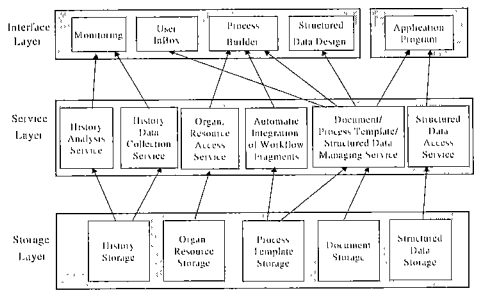


Fig. 2 Layered view of connector facility

3.1 Workflow definition in ICU/COWS

In a connector-oriented workflow system, workflow either can be defined at workflow build time like conventional workflow system or fragments of workflow can be defined at run time. The defined workflow fragments are stored to some place in the form of workflow templates so that they can be exploited for the next workflow definition. The frequency of the usage of the workflow template is registered as basic data for BPR. The last node of the workflow fragments defined at run time should be either the end node of the whole workflow or a connector to the department to which the work item has to be passed. One or more connectors could be connected as the last nodes of a workflow fragment.

For the definition of the workflow fragment of a department, both activity-oriented workflow definition and actor-oriented workflow definition are supported. In activity-oriented workflow definition, which is the standard workflow definition method, workflow is composed of connected activities that have their own attributes respectively. Therefore, in activity-oriented workflow definition, the activity derivation is inevitable and the naming of the activity should be entailed. To the run time definer of the simple transient workflow fragment, the activity derivation and naming process could be cumbersome chores. In actor-oriented workflow definition, the definer only lists the actors that the work items should be handed over. Although the actor-oriented workflow definition has some constraint in defining workflow, it is useful for the non-expert workflow designer to define simple workflow path dynamically. Actually the actor-oriented process definition is adopted and well suited to the decision approval support system. The decision approval path can be determined at run time by designating nodes in the path in the name of actors or roles. Figure 3 shows the typical workflow fragment types that can be defined in ICU/COWS.

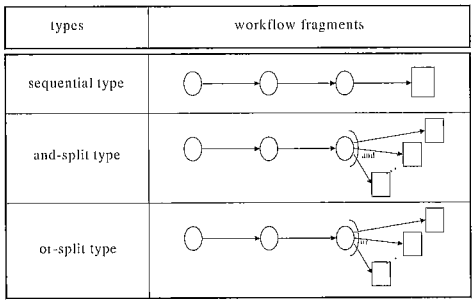


Fig. 3 Three types of workflow fragments

In a connector-oriented workflow system, structured data can be defined at run time similarly to the workflow relevant data definition in a standard workflow system. Once the data are defined they are treated equivalently to the workflow relevant data defined at build time.

3.2 Workflow processing in ICU/COWS

Workflows defined at run time are started by generating a work item and then sending the work item to the first workflow participants. The processing of the work item after the invocation of the job is the same as the normal workflows defined at build time until it reach connectors. Once the work item reach connectors, it is stored in the connectors. Thus the work items received from the preceding departments are cumulated in the connector of a department. For the processing of a work item in the connector, it either can be fetched by the corresponding actor directly or forwarded to

the corresponding actors by the connector manager. Generally, at least one connector manager is assigned to the connector of a department. When the corresponding actor can be decided by the system, the delivery service can be done automatically by the system. Once a work item is delivered to the corresponding actor, he defines the processing path by either selecting the appropriate workflow template from the workflow template storage and starts the workflow or creating a new workflow template and registers the newly defined workflow template before starting the workflow. After the workflow fragment is defined, the process is invoked automatically. This is a typical workflow processing mechanism for a decision approval support system which is very popular in several oriental countries.

3.3 Workflow fragments integration

Integration of the workflow fragments defined by RTPD is very important for the success of the new workflow paradigm. The integrated workflow and statistic data could be an impelling force to BPR and a good starting point of BPR at the same time. Since the connector facility is devised for the integration of the workflow fragments defined at run time from the beginning, the integration of the workflow fragments is rather straightforward in ICU/COWS if the workflow fragments are one of the types allowed in subsection 3.1. Figure 4 shows the integrated form of the five(A, B, C, D, E) workflow fragments. The integrated form is obtained by connecting workflow fragments through the intermediary of connectors. Before we explain the procedure of the integration we introduce workflow fragment description table. The table has information for each workflow fragment.Workflow fragment ID, workflow instance ID, starting connector, end connector list, way of split, merge connector and the level of the workflow fragment are registered in the table. Figure 5 shows the contents of workflow fragment description table for the five workflow fragments.

Before we explain the algorithm we define following functions on the workflow fragments

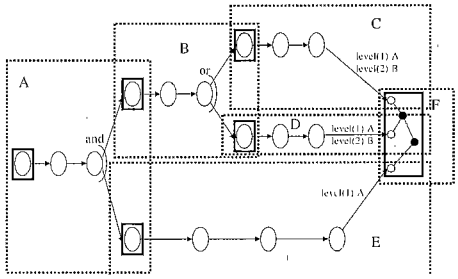


Fig. 4 Integrated form of 5 workflow fragments

denoted by WFfg. Most of the functions can be implemented easily once the workflow fragment description table is given.

type(*WFfg*) : returns the type of the parameter *workflow fragment*.

level(*WFfg*) : returns the level of the parameter *workflow fragment*.

highest_level(*connector*) : returns the highest level of the *workflow fragments* merged into the *connector*.

lowest_level(*connector*) : returns the lowest level of the workflow fragments merged into the *connector*.

WF(*connector*) : returns the *workflow fragment* started from the input parameter *connector*.

start_connector(*WF*) : returns the start connector of the input *WF*.

end_connector(*WF*) : returns the end connector of the sequential input *WF*.

end_connector_list(*WF*) : returns end connectors list of the and-split or or-split input *WF*.

connect_connector(*WF_process*, *connector*) : connect *WF_process* and the input *connector* in the form of tree.

connect_fragment(*WF_process*, *WF*) : connect *WF_process* and the input workflow fragment.

push(*WFfg*) : pushes the parameter workflow fragment *WFfg* into the workflow fragment stack.

pop(*WFfg*) : pops the parameter workflow fragment *WFfg* from the workflow fragment stack and delete the top of the workflow

fragment stack.

The process derivation from the workflow fragments is composed of two stages. In the first stage, the join links of each connector are constructed. When multiple workflow fragments are merged into a connector, the multiple input edges are joined until the final join node is derived. Roughly, for multiple input workflow fragments or join nodes in the same level and they have the same split point, a join node is created so that the workflow fragments are linked to the join node in the second stage. If join nodes are included, the join nodes and the newly generated join node are connected. Next, the level of the newly created join node is set to less than the connected join node by 1 and its appropriate split point is set. The split point of a join node is searched by traversing backwards until it reaches to the workflow fragment whose level is less than that of the join node by 1. Thus the split point of the upper join node of Figure 4 become A, because the level of the join node is 1 and the level of workflow fragment A is 0. This procedure is repeated until the level of the created node is the same with the lowest level of the input workflow fragments. Thus after the execution we can obtain a join link tree for each connector. Workflow fragments are connected to the join trees at the second stage. Algorithm Resolve_Joinlinks(*connector*) shows the procedure that a join link tree is produced for a connector.

In the second stage, a workflow process is constructed by connecting workflow fragments and the join trees. The construction is started from the workflow fragments involving starting connector, which is denoted in the workflow fragments description table. In Figure 5, the starting connector is denoted by underline. If the workflow fragments are sequential type 1 connector is integrated with the workflow fragment and if the workflow fragments are and-split or or-split type multiple connectors are connected with the workflow fragment. Then the workflow fragments started from the attached connector are integrated. This procedure is repeated until no more workflow fragments can be included. The *included_flag*(include() in the algorithm) is tested before integrating a workflow fragment to prevent a workflow fragment is integrated more than once. This situation can happen when multiple workflow fragments are merged into a connector. The algorithm is in Derive_Workflow().

Workflow Fragment ID	Workflow Instance ID	Starting Connector	Next Connector(s)	Way of Split	Merge Connector	Level
A	100	A	(B, E)	AND	1	0
B	100	B	(C, D)	OR	1	1
C	100	C	1		-	2
D	100	D	1			2
E	100	E	1			1

Fig. 5 Workflow fragment description table

Resolve_Joinlinks(*connector*)
Input: A connector.
Output: A join link tree for the connector.

level = *highest_level*(*connector*);
low = *lowest_level*(*connector*);
while(*level* ≥ *low*) {
  for all (fragments fg_1, fg_2,⋯,fg_n and join nodes J_1, J_2,..J_m ∈ *samelevel_fragments*(*level*)){
    partition the fragments and the join nodes into par_1, par_2,⋯, par_k such that each partition has the same split point.
    for all(partitions with multiple elements) {
      create a join node J and link the join nodes of the partition with J.
      set the *level* of J to (*level* - 1).
      decrease the value of *level* by 1.
    }
  }
}

Derive_Workflow()
Input: A set of workflow fragments having the same workflow instance ID.
Output: A workflow process.

```
for all (connectors in the input workflow
fragments)
    resolve_joinlinks(connector);

WF_process =WF(starting_connector);
WF_push(WF(starting_connector));
while ( (current_fragment = WF_pop()) is
    not empty){
    if( type(current_fragment) = "sequential" ) {
      connector = end_connector(current_fragment);
      temp = WF(connector);
      if( not included(temp) ) {
          WF_process =
              onnect_connector(WF_process,
                                  connector);
          WF_process =
              connect_fragment(WF_process, temp);
          set_included(temp);
          WF_push(temp);
       }
    }
    elsief( type(current_fragment) =
       "and-split" or "or-split" ) {
       connector_list =
            end_connector_list(current_fragment);
      while( (connector =
              get_connector(connector_list))
          is not empty){
          temp = WF(connector);
          if( not included(temp) ) {
              WF_process =
                  connect_connector(WF_process,
                                  connector);
              WF_process =
                  connect_fragment(WF_process,
                                  temp);
              set_included(temp)
              WF_push(temp);
          }
      }
    }
}
```

4. Implementation

In this section, we introduce our distributed object oriented workflow system and the implementation of RTPD in the system. We call it CBWS(Connector Based Workflow System) when it accommodates RTPD and is equipped with connector facility. In the implementation of CBWS, we mainly focus on the implementation of RTPD. The integration of induction module in the system is described also.

4.1 CBWS

Connector facility is one of special features to support RTPD of workflow system. Thus connector facility does not cover all the generic functions workflow system is supposed to provide. In this paper, we do not consider the rest of functions and services of workflow systems should be provided and we only explain the way of RTPD is implemented.

Connector facilities and RTPD can be implemented in several different ways. One is to extend the functions and modules of conventional workflow system. In that case, the base workflow system should be flexible enough to accommodate the extensions easily.

The other way is to develop new workflow system from scratch based on the concepts. The ordinary functions and services are implemented together during the development. But this approach takes a lot of efforts until the service can be used. Actually we choose the first approach because we already have very flexible distributed object based workflow system.

Meanwhile the connector facility can be implemented relatively easily by modifying the organizational inbox in a way. A work item arrived at an and-joining point may has to wait another corresponding work item to arrive and is merged into one work item to be delivered to an appropriate actor. But for the support of RTPD, more fundamental modification to workflow engine is required. In the following section we briefly introduce the base workflow system and its

extensions to accommodate RTPD.

4.2 ICU/COWS

ICU/COWS is a distributed object oriented workflow system developed in CORBA environment. Each component of the system is in the form of CORBA object and accessed through CORBA naming services. The system has unique features in that each workflow activity is controlled by an object instance named TMI that is created by specially devised mechanisms. So it can be viewed instance based workflow system also. Before we explain the mechanisms, we explain each component of the system briefly.

The components of ICU/COWS include TMIF(Task Managing Instance Factory), GTMIG(Global Task Managing Instance Generator), Simulator, Process Builder, Admin/Monitoring Service, and Worklist Handler. In the following sections, we describe the details of some core modules and mechanisms. Figure 6 shows the software architecture of ICU/COWS.

4.2.1 TMI and GTMI

Before we explain the TMI and GTMI(Global Task Managing Instance), we introduce the ExecObject which is the abstract object for TMI and GTMI object. ExecObject is composed of common attributes, status information, and methods for both TMI and GTMI objects. Operations to

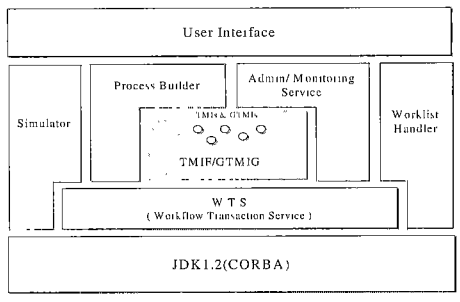


Fig. 6 Software architecture of ICU/COWS

create, manage, and access to history information is also included in the ExecObject methods.

TMI of an activity managing the task of the activity is created from the TMI object. It either sends a work item to a worklist handler or invokes an application through application agent. In between application and TMI, application agent relays workflow relevant data via TMI. TMI also monitors the status of the invoked tasks by communication with worklist handlers or application agents. When a task is completed the TMI sends the start event to the next TMI and the TMI which receives the signal of the completion of its former task starts its task. In this way, control is transmitted as defined at process build time.

GTMI controls the processing of global process instance from the creation of a process instance to the end of the process instance. When all the TMIs are created for the process, GTMI starts the process instance by triggering the first TMI. During the processing it either receives status reports from the TMIs or suspends TMIs transiently to handle the requests from the administrator such as dynamic reconfiguration. Although a TMI has to report its status to its GTMI, TMI can continue its execution even if the GTMI crashes because it does not check whether the GTMI has received its report or not. This approach is effective to achieve availability in a distributed environment where the network disconnection is frequent. When the last TMI reports its end of processing, the GTMI erases all the created TMIs and de-allocates spaces allocated for the process instance including itself.

4.2.2 Generation and Working Mechanisms of TMI and GTMI

A distributed server is equipped with one GTMIG and one or more TMIFs. When a client asks GTMIG to generate a process instance, GTMIG creates a GTMI for the process execution. Once a GTMI is created, it asks TMIFs to generate all the TMIs for the process instance through GTMIG. We call it setup phase of COWS. Thus TMIs are generated initially when client requests a new process instance.

When the request to start a process instance from a client, GTMIG receives the request(1). Then GTMIG determines the TMIF to create a GTMI(2,

3, 4) and creates a GTMI(5) and TMIs. Once all the TMIs are created, the GTMI asks the first TMI to start the work(6). If first task is user interactive, TMI send a work item to worklist handler. If it is automatic task, it invokes application agent connected to application program. When the end of the task is signaled to the first TMI, it asks the second TMI to start its task(7). When the last TMI is finished, it informs the GTMI of its' work done(10). Figure 7 illustrates this process in sequence numbers.

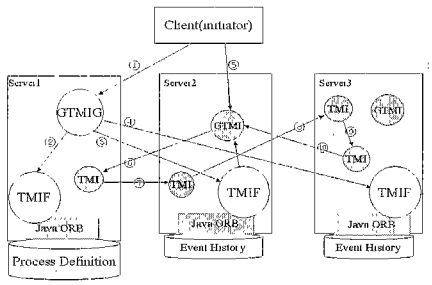


Fig. 7 Sequences of TMI and GTMI creations and operations

Since numerous TMIs are generated and the TMIs may be created on different servers, it asks the creation to the TMIF resident on the same site as the generated TMIs. To the GTMI, local TMIFs and remote TMIFs are viewed equivalently and they are invoked in the same way. So the workflow system operates in a fully distributed fashion. The site where a TMI is to be created is determined either by the user directives or considering the system configurations. When one server is down, the GTMIG searches alternative servers and uses the selected server on behalf of the crashed server. In this way, the whole system can maintain high system availability irrespective of a system failure.

4.3 RTPD Accommodation

4.3.1 Generic Views

To accommodate run time process definition capability, a workflow management system should provide several additional functions. First, it should allow workflow fragments to be defined and handled at run time and to be registered and managed for later usage. This is disregarded or overlooked in dynamic reconfiguration. Second, WFMS should provide automatic integration and visualization functions for workflow fragments. Automatically integrated workflow processes can be used for BPR by refining the processes. Third, WFMS should allow the process instances generated by RTPD to be seamlessly integrated with its monitoring and administration modules. For instance, the workflow instances generated by the RTPD facility can be monitored and managed in the same way as the workflow instances generated from the workflow templates defined at build time. Fourth, the functions supplied at process build time by a process modeling tool have to be provided in a form adapted to run time process definition appropriately. Fifth, the RTPD facility should be connected to the run time client user interfaces and can be invoked easily and interactively.

4.3.2 RTPD Accommodation in ICU/COWS

In this section we only focus on the extensions of ICU/COWS to implement run time creation of TMIs and GTMIs and to provide consistent handles for managing and monitoring the workflow processes. The other facilities can be implemented relatively easily.

As the TMI and GTMI creation mechanisms are very flexible and dynamic, only a few things need to be changed for the accommodation. Once a workflow fragment is defined by a participant then it is handled in the same way as the ordinary workflow templates. The defined workflow fragment is saved somewhere in the database of workflow system and the corresponding TMIs and GTMI are created using the mechanism described in the section 4.2.2. Thus there is only one GTMI for each workflow fragment.

This approach does not have any problem in processing workflow fragments but it provokes several problems when controlling and monitoring a whole process. Several GTMIs are involved in consisting of a whole process. Each GTMI should

be controlled separately and another service module is required to provide the integrated view on the process.

To resolve this problem we adopt the hierarchical structure of GTMIs. That is, GTMIs may have a common upper level GTMI. GTMIs of workflow fragments composing a workflow process share the same GTMI. We call the upper level GTMI Global GTMI(GGTMI) and the lower level GTMI Local GTMI(LGTMI). Figure 8 depicts this relation with three workflow fragments. Here workflow fragment may be considered as subprocess.

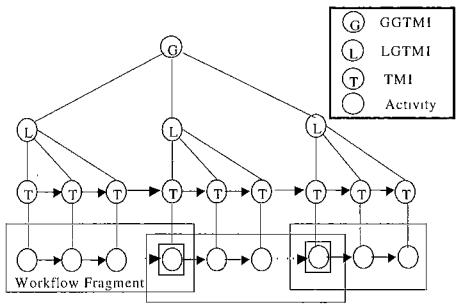


Fig. 8 GTMI hierachy

To implement the hierarchical structure of GTMIs, we slightly extend the creating mechanism of GTMI. When the creation of a workflow instance of the first workflow fragment composing a workflow process is requested, two GTMIs are created. One is GGTMI and the other is LGTMI. After the GGTMI is created, the creation of workflow instances of other workflow fragments is conducted by creating LGTMIs and registering the corresponding GGTMI to LGTMI and vice versa. In monitoring service, we can refer to the GGTMI for the monitoring of a process situation. GGTMI reports the situation by asking and integrating the local situations of all the active LGTMIs.

4.4 Integration of Workflow Model Induction Module

To utilize the functions of the workflow model induction, the workflow induction module has to be integrated with the connector based workflow system and process modeling tool. The connector based workflow system provides interfaces to define workflow fragments at run time, and collects the defined workflow fragments into the workflow fragments table. Workflow instance integration module and workflow model induction module create workflow templates and store them at the workflow template table which is connected with the process modeling tool.

Process designers or BPR experts refer to the induced workflow model in designing workflow models via the process modeling tool directly. They can just modify it in designing a workflow model rather to draw it from the scratch. Figure 9 shows the connections among them. The connector based workflow system in the Figure represents the one implemented by extending our connector-oriented workflow system [5]. The process modeling tool is also extended to provide interfaces to access the induced workflow models. The numbers in the circle denote the procedure from the definition of the workflow fragments to the reference by the process designers or BPR experts of the induced workflow models.

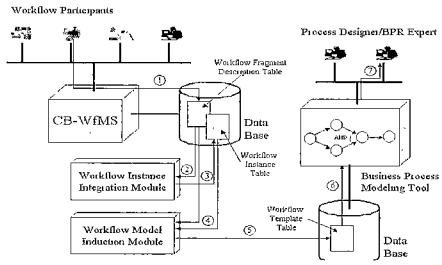


Fig. 9 Connections among workflow management system, induction modules and process modeling tool

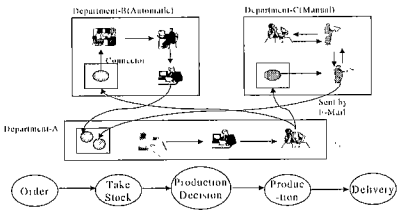
5. Application

In this section, we explain how the run time process definition can be applied to the following situations. We assume that during the workflow execution of Figure 10(a), the participant of the

production decision activity has found that the decision of production should be approved by other departments as well as his department. Thus the participant has to define new approval path at run time. Figure 10(b) shows how the procedure is handled in ICU/COWS. Firstly the participant defines the approval path from the participant to the head of the department via his senior member as shown

in the department-A block in Figure 10(b). Note that the last nodes of the definition should be the connectors of his or other departments. After the approval path is defined he starts the work item along the approval path.

(a) Original business flow defined at build time



(b) Combined business flow defined at compile and run time

Fig. 10 Application of the connector-oriented workflow system to an example workflow application

Once the approval process of the participant's department is completed the work item is delivered to the connectors of the designated departments. The delivered work item is pushed to the appropriate person in the department by the connector manager of the department or pulled by the person directly. Once the work item is delivered to the corresponding person, the same steps as those the participant of department-A has taken are taken. Note that, if the department is a manual department, the work item might be processed manually and the work item is delivered to the connector of the target department through e-mail. Thus the connector should be registered as an entry of an e-mail system and the e-mail inbox be integrated in the connector.

After all the approvals have been received through the newly defined approval paths, the participant of the activity of the production decision can continue the workflow process as defined. Consequently, in Figure 10 we showed how the predefined workflow and the newly defined workflow defined at run time can be integrated in connector-oriented workflow system in the mixed situation of automated and manual departments.

6. Related Works

Although numerous works[1][2][10][26] have been done on adaptive workflow management systems, relatively few works have been performed on automatic induction of workflow models based on the notion of run time process definition. Kradolfer[25] offers a method for dynamic workflow evalution. It supports modification of workflow definitions with versoning and classification of modification operations. It can keep the modification history and migrate an workflow instance to a modified workflow definition.

However, It differs from the proposed approach in that it does not consider the discovery of workflow process from modified definitions. ADEPTflex[14] presents a formal foundation supporting dynamic structured changes of running workflow instance. It considers neither schema evolution nor process induction. ADEPTflex allows modification of running workflow instance while it keeps correctness and consistency criteria.

Agrawal[14] induces an workflow model from the history of workflow instances while we induce a workflow instance from a set of workflow fragments. A machine learning approach to acquire a workflow model is used in RAP[15] and Herbst [15][18][19]. RAP uses dialog-based learning to induce a workflow model by observing structured e-mails. Herbst proposes to integrate a machine learning component into a workflow management system by extending hidden markov models[20][22]. They define four problem classes according to the

characteristics of the target a workflow model. But the unit of a workflow instance is too primitive to induce complete workflow model. Wolf[27] proposes an event-data analysis framework for process discovery called Balboa. Balboa analyzes the event data collected from executing processes to find behavioral patterns of processes. It uses three methods for process discovery that are included in the algorithmic technology or statistical technology called KTail, RNet and Markov. Although the method is developed in different context from workflow area, it pursues the same goal and the technique can be utilized in automatic induction of workflow models. But the kind of events and the ways of event catching should be changed appropriatly to be applied in workflow context.

The idea of the utilization of organizational member's behavior in workflow process induction can be found in [23], [28] and [29]. Although we share the idea with them to some degree, our work differs in that the integration of organizational learning and workflow management system is very important for the real success of workflow management systems. WorkBrain[23] introduces an approach for an evolutionary organizational memory based workflow system. The system uses work cases of organizational memory to find a business process. It supports various operations for building workflow management tasks. Users can modify workflow instances and WorkBrain saves these changes which will be used later to create a new workflow building block by users. WorkBrain differs from the proposed approach in that it does not describe a concrete algorithm to derive a new process from workcases.

7. Conclusion

Many workflows of large scale enterprise is still in non-automated state. This is partly because of weak intention of top managers but the restricted capability of conventional workflow management system is also another critical hindrance for the automation. Usually the workflows of large scale enterprise are in the mixed type of workflows and the run time process definition is very important for them to be processed automatically. Although dynamic reconfiguration facility of conventional workflow management system can cover some situation, it has limitation to fully support run time process definition.

Meanwhile in the conventional workflow paradigm, the whole workflow should be defined before it to be executed. But for a workflow system to cover much wider area of workflow, the new workflow paradigm in which a workflow can be started after only a part of workflow is defined should be supported by workflow management system. The undefined part of workflows could be defined at run time appropriately according to situations. Thus we can expect much wider coverage of workflows by workflow management system equipped with run time process definition facilities. Which is another background that workflow management system should support run time process definition.

In this paper we have proposed the connector facility as an implementation method of run time process definition for workflow management system. Three types of workflow fragments are allowed for the definition at run time. We showed a workflow can be derived automatically once workflow fragments in the form of aforementioned three types are given and the algorithm of deriving a workflow from workflow fragments are illustrated. This implies that workflow management system is merely not the means to automate processes defined through BPR. Rather it can be used also in the early stage of BPR in which even process is not defined at all to help to derive current processes. Which can diminish the cost of BPR drastically and makes the transition from non-automated state of a company to fully automated state smooth. This is because the usage of run time process definition of workflow management system could be a intermediary state between non-automated state and fully automated state of a company.

During the integration of the connector facility to

ICU/COWS where TMI is in the form of object instance and a GTMI controls all the TMIs of a workflow instance, we have found the architecture of ICU/COWS is very flexible. We can create the TMIs at run time easily and we can treat both the TMIs created at build time and run time in the same manner once they share the same GTMI.

References

[1] P. Lawrence. Workflow HandBook. John Wiley & Sons Ltd. (1997).

[2] L. Fischer.The Workflow Paradigm. Future Strategies, Inc. (1996).

[3] ORBWork: A Distributed CORBA-based Engine for the METEOR Wokflow Management System. University of Georgia, Athens, GA, http://LSDIS.cs.uga.edu/.

[4] C. Ellis, C. Maltzahn. Chautauqua: A Flexible Workflow System. Proc. of the 30th HICSS Conference, (January. 1997).

[5] D. S. Han, J. Y. Shim, C. S. Yu. ICU/COWS: A Distributed Transactional Workflow System Supporting Multiple Workflow Types. IEICE Transactions on Information and Systems Vol. E83-D, No. 7, July 2000.

[6] Nortel & University of Newcastle upon Tyne. Workflow Management Facility Specification Revised Submission, OMG Document Number: bom/98-03-01 (1998).

[7] Workflow Management Coalition Specification Document. The Workflow Reference Model. Version 1.1 (November 1994).

[8] Joint Submitters. Workflow Management Facility. Revised Submission, OMG Document Number: bom/98-06-07, July 4, 1998.

[9] Z. Yang and K. Duddy. CORBA: A Platform for Distributed Object Computing. ACM Operating System Review, Vol. 30, No. 2. Pages 4-31 (1996).

[10] D. Georgakopoulos, M. Hornick, A. Steth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed Parallel Databases, Klewer Academic Publishers, Volume 3, Number 2, pp. 119-154 (1995).

[11] C. Ellis, K. Keddara, G. Rozenberg. Dynamic Change within Workflow Systems. Proceedings of the ACM SIGOIS Conference on Organizational Computing Systems, Milpitas, CA., pp. 10-21 (1995).

[12] M. Reichert, P. Dadam. A Framework for Dynamic Changes in Workflow Management System. Proceeding of DEXA'97 (1997).

[13] M. Reichert and P. Dadam. ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems 10(2), 1998

[14] R. Agrawal, D. Dunopulos. F. Leymann. Mining Process Models from Workflow Logs, In Proc. of the 6th International Conference on Extending Database Technology(EDBT), 1998.

[15] S. Bocionek, T. M. Mitchell. Office Automation Systems that are Programmed by their Users. In 23. Jahrestagung der Gesellshaft fur Informatik, pp. 214-219, Berlin, Germany, 1993, Springer-Verlag.

[16] Clarence A. Ellis, Keddara K, Rozenberg G.. Dynamic Change within Workflow Systems. In Proc. of the ACM Conference on Organizational Computing Systems, pp. 10-21, 1995.

[17] Y. Han, A. Sheth. On Adaptive Workflow Modeling. In Proc. of the 4th International Conference on Information Systems Analysis and Synthesis, pp. 108-116, Orlando, Florida, 1998.

[18] J. Herbst, D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. In Proc. of the 9th International Workshop on Database and Expert Systems Applications, pp. 745-752, IEEE, 1998.

[19] J. Herbst, D. Karagiannis. An Inductive Approach to the Acquisition and Adaptation of Workflow Models. In Proc. of the IJCAI'99 Workshop on Intelligent Workflow and Process Management, Stockholm, Sweden, 1999.

[20] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Proc. of IEEE, 77(2):257-285, 1989.

[21] S. K. Shrivastava and S.M. Wheater. Architectural Support for Dynamic Reconfiguration of Large Scale Distributed Application. The 4th International Conference on Configurable Distributed Systems(CDS'98), Annapolis, Maryland, USA, May 4-6, 1998.

[22] A. Stolcke, S. Omohundro. Best-First Model Merging for Hidden Markov Model Induction. Technical Report, TR-94-003, International Computer Science Institute(ICSI), 1994.

[23] C. Wargitsch. WorkBrain: Merging Organizational Memory and Workflow Management Systems. In Workshop of Knowledge Based Systems for Knowledge Management in Enterprises at the 21st annual German Conference on AI(KI-97), pp. 214-219, Kaiserslautern, Germany, 1997.

[24] W. M. P. van der Aalst, T. Basten, H. Verbeek, P. Verkoulen, M. Voorhoeve. Adaptive Workflow: On the Interplay between Flexibility and Support. In J. Filipe and J. Cordeiro(editors), Proc. of the 1st International Conference on Enterprise Information Systems, pp. 353-360, Setubal, Portugal, 1999.
[25] Markus Kradolfer, Andreas Geppert. Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration. Proceedings of Fourth IECIS International Conference on Cooperative Information Systems, pp.104-114, Edinburgh, Scotland, 2-4 September, 1999
[26] Amit Sheth. From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Coordination and Collaboration. Proc. Workshop on Workflows inScientific and Engineering Applications, IEEE Computer Soc. Press, Los Alamitos, Calif., 1997
[27] Jonathan E. Cook and Alexander L. Wolf. Discovery and Validation of Processes. NSF Workshop on Workflow and Process Automation in Information Systems, Athens, Georgia, May 1996.
[28] Mark S. Ackermann. Augmenting the Organizational Memory: A Field Study of Answer Garden. Proceedings ACM Conference on Computer Supported Cooperative Work, Chapel Hil, October, 1994
[29] M. Berger, E. Ellmer, D. Mcrkl. A Learning Component for Workflow Manangement Systems. Proc. 31st Annual Hawaii International Conference on System Sciences(HICSS-31).

한 동 수
정보과학회논문지 : 데이타이스
제 27 권 제 2 호 참조

심 재 용
정보과학회논문지 : 데이타이스
제 27 권 제 2 호 참조