INTEROPERABILITY WITH WORLD WIDE WORKFLOWS

Herbert Groiss, Johann Eder

Institut für Informatik
Universität Klagenfurt
Klagenfurt, AUSTRIA
e-mail: {herb,eder}@ifi.uni-klu.ac.at

ABSTRACT

In this paper we present a workflow system which is innovative in three respects: First, we use active databases for the implementation of a workflow server. Second, WWW browsers as clients allows platform independent worldwide access to the workflow management system. Finally, we show how interaction between workflow servers for communication between different workflow systems can be achieved with a simple extension to this architecture.

1 INTRODUCTION

Workflow systems are a key technology for supporting the management of processes as it is requested by modern organizational approaches. Workflow management systems support the definition, execution, controlling, and documentation of business processes.

Several products of workflow management systems are already on the market with a different set of features and different degrees of support (IBM, 1994; Georgakopoulos et al., 1995). The Workflow Management Coalition was founded to define standards for terminology and interfaces of workflow management systems (Workflow Management Coalition, 1994). Although there is an increasing number of success stories, there are still several technical problems to solve.

One of the urgent issues is to increase the openness and interoperability of workflow systems in several directions:

- Companies spread over many distributed locations need to be integrated. Some applications link together more than 4000 sites, geographically distributed, using heterogeneous platforms (Mohan et al., 1995).
- Interfaces to users outside the organizations must be available. A business process typically starts outside of an organization, for example by submitting requests or orders.
- Workflow systems of different companies should be able to cooperate in business processes.

We present the prototype workflow management system Panta Rhei which cares of the demands outlined above. In our approach the definition of workflows relies on the formflow metaphor (Hämmäinen et al., 1990). Within a single organization, the coordination between the tasks a business process consists of is achieved by forwarding of forms which contain the data necessary for performing a task.

The communication between two (or more) enterprises is typically handled through exchange of documents (orders, bills, ...). The receiving of a document initiates or continues a business process at the receiver side. Sending a document to another enterprise is a (partial) result of a business process. Therefore, we can also use the formflow metaphor for the cooperation of workflows in different organizations. In good software engineering tradition (Yourdon and Constantine, 1979) we aim at a low coupling level between tasks, between workflows and between enterprises.

In our prototype system all data of process schemas and instances are kept in a database. The definition of the formflows are mapped to the triggers of an active database system which controls the execution of workflows. This architecture has several advantages - among them are the ease of implementing a workflow server and the recoverability of dynamic business processes in case of hard- or software failures.

The Internet and the World Wide Web are widely available and used. Many users are familiar with these media. Employing these systems as front end to a workflow management system and as communication vehicle for the exchange of forms (documents) we can easily meet the demands outlined above. WWW-servers and clients are available on various hard- and software platforms. Therefore, they can be used as integration platform for workflows in large and heterogeneous environments. Furthermore, they offer the opportunity to open workflows to users outside an organization (e.g. to customers). There are already simple open business processes available on the Internet, like ordering pizzas or books, subscribe to a mailing list, etc. In our approach the sending of an order form would initiate a business process at the receivers side. The form is transmitted to the receiver via WWW and the Internet. A WWW - database gateway stores the form in the database and this triggers the execution of succeeding tasks. In a similar way we achieve the cooperation between workflow systems in different organizations. The workflow systems communicate by exchanging forms via the Internet.

In the next section we introduce our workflow description language, its graphical notation and the execution model. Then we present the implementation using active databases. Section 4 shows how WWW browsers can be used as clients for the workflow system. The concept and implementation of cooperation between workflow servers is described in section 5. Finally we draw some conclusions and outline ongoing research.

2 WORKFLOW DESCRIPTION - WDL

Workflows are compound activities, which describe the execution of *tasks* by different *agents*. An agent is either a concrete user or a program, which performs the data manipulation, or a role as placeholder for a set of agents. *Forms* are the data containers, which hold the data for manipulation and for communication between the tasks. The transmission of forms between two activities is specified with *flows*.

In the following we present the Workflow Description Language WDL (Eder et al., 1994), which allows the graphical representation of workflows with a so-called process diagram.

Fig.1 shows such a diagram for the process of a paper review for a conference. This process is typical for the distributed nature of some business processes. Typically no two agents are in the same organization. An author sends a paper to the program committee chair, who sends copies of it together with a review form to the referees. When the PC chair has received all referee reports he makes an evaluation and sends a summary to the authors. The revised versions of the accepted papers are finally sent to the publisher.

The tasks are represented by rectangles, inside the rectangle the name of the task and the agent (the user or role performing the task) is written (the name of the agent is enclosed between brackets). If the task is processed automatically - without an user interaction - the pseudo-user SYSTEM is specified. This allows the definition of arbitrary programs for manipulation of the data and therefore the integration of other application programs into the workflow. Sometimes it is useful to define the user dynamically, i.e. send it to the task as content of a field in a form. In this case we write DYNAMIC into the user field.

Note that the concept of dynamic users is very powerful, for example e-mail can be modeled if the user can write the field of a form where the agent of the next task is read from. The process diagram of e-mail is a task with a flow starting and ending at this task.

The flows are represented through arrows. Each arrow represents that one or more forms are sent from one task to a successor task. These flows can be combined in several ways:

1. Input side: a) no precondition: That means the task is activated if one of the input forms arrives.

   b) a boolean expression together with an optional predicate: We define explicitly the valid combinations of the input forms (e.g., f1 AND f2, that means forms f1 and f2 must be available before activating the task) and a predicate for synchronizing the forms (e.g., f1 AND f2 [f1.name = f2.name] means forms f1 and f2 must be available and reference the same name).

   c) a synchronization point: A form can be sent to more than one successor task for parallel manipulation. At the end of such a parallel processing the synchronization point is only passed if each of these parallel tasks are finished.

2. Output side:

   After completing the task each output flow transports its forms to the specified successor task (if an optional condition is valid). Consider the following special concepts:

   a) disjunction: The actual form is either sent to task A or task B depending on the condition specified by the flows. Using this concept we can model conditional flows.

   b) a form is sent to task A and task B for parallel manipulation: This is the counterpart of the synchronization point introduced in the above paragraph. For modeling parallel manipulation a form 'splitting' at the begin and a synchronization point at the end has to be defined.

Though the process diagrams are very illustrative in showing what is going on, some additional information have to be expressed for compilation of the description to an executable workflow. For example the types of the fields in the forms have to be specified.

1. General information for the workflow:

   a) associated users and roles: The process diagram just shows the participating users and roles. In addition it is necessary to define all the users participating in a workflow and the association of roles to users.

   b) structure of the involved forms: Again the process diagram just shows the flow of the forms without defining the structure of the form. A form consists of fields each having a type. We allow atomic types (string, number, boolean and character), the type table (a collection of tuples of atomic types), as well as references to external files. Moreover, the appearance of a form in the user interface has to be defined.

2. Additional information for tasks are

   a) postconditions: You can explicitly define some postconditions to enforce a valid state. The task can be successfully completed only if the postconditions are fulfilled.

   b) procedure: A before-procedure and an after-procedure can be specified for execution before activating or after completing a task respectively. The procedures and the post-condition are optional.

   c) selection criterion: Specifying a role as task performer requires the definition of a selection criterion. This criterion is used for the assignment of the task to a concrete user during the execution. Possible criteria are: choose user with minimal workload, choose user randomly, etc.

   d) access structure: It is possible to specify which fields of a form a task can read or change.

We have not defined an exact syntax, how the preconditions, postconditions and the procedures are specified. This is left unspecified, because it depends on the concrete implementation, i.e. in our case on the data manipulation language of the database management system.

2.1 EXECUTION MODEL

A WDL process description defines when and under which conditions a form is transported from one task to a successor task. What is done within a task is not specified. After such a form manipulation in a task A is finished, the workflow system executes the following steps:

1. the optional after-procedure of task A is processed.

2. The postconditions of task A are evaluated. If the postconditions are fulfilled, the forms manipulated by this task are marked as processed and the task is finished, in the other case the task gets an error message.

3. Every output flow of task A is checked and if the flow condition is met, the form is sent to the successor task and gets the status pending.

4. The preconditions of every successor task are evaluated. If all preconditions of a task are met the task is ready.

5. If there is a task ready, the next step is the assignment of a user to the task if the specified agent of the task is a role. The selection criterion is evaluated and a concrete user is assigned to the task.

6. Next the (optional) before-procedure is started.

7. The user interface of the user assigned to the successor task gets now a signal that the task can be started.

3 ACTIVE DATABASE AS WORKFLOW SERVER

One major contribution of our approach is the usage of active databases to implement the workflow machine. Active databases are well suited for applications which are inherently data driven or event driven (for an introduction into the field of active databases refer to (Dayal, 1988; **?**)). These systems extend conventional (passive) databases with production rules. They allow the specification of actions which are executed automatically whenever certain events occur and certain conditions are satisfied. The specification of Event, Condition and Actions is done declaratively with so-called ECA-rules.

Each database access from an user or an application program (insert, update, delete, select) is seen as an event, which can trigger the application of a rule. If a rule is triggered, the conditions of the rule are evaluated. If they are satisfied, the actions of the rule are applied. Conditions are descriptions of database states, actions are operations, which can modify the database or start external procedures.

Using this mechanism we can define rules which react automatically on the change of the status of some forms and make an action according to the definition of the workflow.

The structure and content of the forms as well as the information about users and roles are maintained in database tables. Additional fields are needed for administrative and dynamic information: the holder of the form, the task which currently has access to it, and the status (pending, active, etc.). The rules are automatically generated from the declarative descriptions of the tasks and flows by the WDL compiler. Therefore, the active database management system is the workflow server and has the functionality described in the process specification.

Mainly, the rules react on changes of the status fields of the forms. For example, when a task is finished it changes the status of the processed forms from active to processed. This event fires a rule which runs the post-procedure and changes the status of the forms again.

In this way a chain of rule applications is initiated, whenever a task is completed. In analogy to the steps described above, the description of a workflow is translated into several groups of rules.

For each flow one rule is generated, triggering when a task is completed, i.e. after the satisfaction of the postcondition. This corresponds to the third step of the above execution model. The following rule specifies a flow of a form of type *form_i* from *task_A* to *task_B*, where the form is sent if the condition *flow-condition* is met:

```
create trigger flow_n_step3 on
   form_i
after update status
when new.status='finished'
and form.type=form_j and form.task
   = task_A and flow-condition
then update new set task = task_B;
```

The rule fires on changes of the status field in the table *form_i*. The condition is met if the new value of the status is 'finished'. In this case the task field of the form is set to the successor task.

Like in the above example, the rules are built from fixed templates into which the information from the process specification is filled in, e.g. from-task, to-task, form, and flow-condition.

For each task a set of rules is generated for performing the precondition, before-procedure, user assignment, after-procedure, and postcondition. The workflow manager then simply consists of all the rules resulting from the compilation of WDL workflow specifications. All other necessary features are already provided by the database management system.

What are the advantages of using active databases as base technology for implementing workflow systems?

- All dynamic information like the (dynamic) status of processes, documents, etc. are mapped to the database and maintained within a database system. Thus the capabilities of database systems like safety, authorizations, and most important recovery are immediately available. In particular, in the case of system crashes, the recovery mechanism of the database also recovers the dynamic state of all processes.

- Workflow processes should provide a high degree of concurrent execution to decrease turn-around times. The transaction mechanism permits to increase concurrency in a safe way. The concurrency control system of the database can directly be used and it is not necessary to re-implement an additional one for the workflow machine.

- If *active* databases are employed, the database is not only the blackboard for the workflow scheduler and the workflow processes, but it *is* rather the workflow machine itself. In particular, the scheduler and the agents no longer have to poll the database whether the preconditions of some process are fulfilled, creating an unnecessary high workload or reducing responsiveness. Previous work has shown that a central scheduler has advantages over sending or polling strategies (Eder et al., 1986).

The described architecture of the workflow server causes loose coupling between server and client: The forms are loaded to the client, the data manipulation happens off-line, and finally the modified data are retransmitted to the server. This kind of interaction allows us to use WWW browsers as client applications, what is described in the next section.

4 WWW BROWSERS AS CLIENTS

There are two possible ways to interact with a workflow system: (1) as unregistered user, (2) as registered user. In the first case a user has limited possibilities to interact with the workflow system: he can initiate a new task, fill in some forms, submit it, and leave his e-mail for later contact. This is a frequently used mode of interaction. The authors in our conference example interact in this way with the review process.

In the second case the user receives a list of tasks currently in the worklist, can now manipulate an associated form and complete the tasks. Note that this mode is not restricted to users inside an organization, the referees in the above example can interact in this way. Both ways of interaction can be realized easily using WWW clients.

World Wide Web is a wide-area information retrieval initiative aiming to give universal access to a large universe of documents. It has bee developed by the European Laboratory for Particle Physics (CERN) and relies on the Hypertext Transfer Protocol (HTTP), which defines the communication between a web client (browser) and the server. Popular web browsers are Netscape (from Netscape Communications Corporation) or Mosaic from the National Center for Supercomputer Applications (NCSA). This browsers support the representation of text, graphic, and audio data. The representation language is is called HyperText Markup Language (HTML).

One major feature of this language are fill-out forms, which allows users to return information to the WWW server. This feature facilitates the usage of web browsers as workflow clients: When a user connects to the WWW-server for performing data manipulation in some form, the server reads the form definition from the database, fills in the actual values of the form fields and sends it to the client. When the user has finished the data manipulation he sends the new values back to the server by pressing a submit button on the form. These values are then saved in the server database.

Fig. 2 shows the appearance of a browser with a form handling the paper review-task of the process of reviewing papers for a conference.

Using Web browsers as clients has several advantages:

1. support of different platforms: It would need a lot of manpower to write and maintain clients for different platforms and it is impossible to rival with the developers of WWW browsers in porting clients to other machines or operating systems. Moreover, on different platforms users can work with their favorite browser.

2. users are familiar with the handling: Web browsers are widely used for getting information form the net, therefore most potential workflow users know them; the browsers are widely available, some of them even public domain (e.g. Mosaic) or part of the operating system (e.g. in OS/2 Warp). Online Help and documentation is available. The switching from performing a workflow task to other work with the interface is easy.

3. mobile computing: The support of different platforms and the loose coupling between the server and the client allows the distributed execution of a task. A user can start the execution of a task on his computer in the office, save the forms to a floppy disk, works off-line at home or on a journey and reloads the results from another computer.

4. secure transmission: Some browsers already support encrypted transmission of data over the net, this feature is indisputable when using the Internet for transmitting secret information.

5. monitoring: different views on the data are necessary for monitoring what's going on, the user should be able to see what happened with "his" forms. A system operator needs to monitor the state of all processes, the individual worklists, etc. Using databases for storing the process information and WWW browsers as interface allows a simple implementation of such add-ons.

6. simple implementation: Additionally to the advantage of no implementation on the client side, the implementation on the server side is easy, because many commercial database management systems already offer WWW gateways.

5 COMMUNICATION BETWEEN WORKFLOW SERVERS

When a process is initiated in our workflow system, the client connects to the server and gets the initial version of the form(s), make some modification on them and send them back to the server.

It is possible to omit the first connection to the server and let the client create the forms. After the data manipulation the client sends the form(s) to the workflow server. This allows to initiate a workflow by sending a form to the server without further interaction.

On the other hand we can enable a workflow server to send documents to other servers. We simply allow a workflow server as agent in tasks. If such task is reached when executing a workflow the server sends the task to the workflow server on the other machine. The only thing to check is the compatibility of the exchanged forms.

In this way arbitrary communication between two or more processes is possible. A task performed by another workflow system can either initiate or continue a remote process.

Example An acquisition usually needs interaction between at least two companies - the buying and the selling one. Fig. 3 shows a part of an acquisition process. Both companies have defined a workflow for this purpose: The buyer first collects his requirements and sends a request for an order to the other company. Depending on the reply it receives, further actions are triggered. The selling company reacts on the receiving of an request, performs some tasks and sends an offer to the customer.
The interaction works when the workflow of the buyer has a task where the agent is the workflow server of the other company. The workflow on the seller side can be initiated from outside and has a task where it sends the order back to the buyer.

The implementation of the described mechanism in our workflow management system is simple and consists of two procedures:

1. *send a form to another server:* Whenever a process execution reaches a task with another system as agent, the server transmits the form to the other system and either completes the task (no output flow) or puts the task on the worklist for the other server. The transfer protocol can be the same as for client server connections, so that the other server needs not to distinguish between a connection to a server or a client.

   However, if an answer is expected (like in the above example), the form sent should contain information that allows the other system to reply.

2. *receive a form:* The other server must check whether the received forms match the definition and either initiate a new process or finish the appropriate task. This functionality is also necessary if we allow clients the creation of forms.

   When the system receives a form, on which it can not react automatically (no workflow specified or form type not recognized) it is forwarded to an employee, i.e. it appears on the worklist as special exceptional workflow.

Based on this mechanism we can use the workflow system as an universal communication tool: A large part of the correspondence can run through the workflow system if a variety of forms is offered to the communication partners. Some of them can initiate a process, others may be treated like e-mail where the recipient can be given directly or is found using the document type and user roles.

Moreover, new workflows can be integrated without notifying the business partners. Because the communication runs over forms, the other party needs not to know how the received forms are treated internally.

Together with the use of WWW browsers this technique allows easy cooperation between the workflow systems of different companies. Only the address of the workflow server, what types of forms it can handle, and for which forms it will reply with an answer have to be published.

6 CONCLUSIONS

We presented an architecture of an open workflow management system which is based on active databases as server technology and the use of WWW-clients and the Internet for worldwide access and platform independence. We extended the formflow approach to open for interoperability of workflows of different organization. As a proof of our concept we implemented the prototype Panta Rhei with Oracle DBMS and the Oracle-WWW Interface (WOW).

Since the cooperation of workflow systems is based on the exchange of documents we can further increase the possibilities for interoperation by using standards for document interchange. The idea of electronic data interchange (EDI) is not a new one and the necessity to standardize the structure of documents has been recognized. EDIFACT (Berge, 1994) defines the type and structure of a wide variety of forms exchanged in business processes. Additionally, the encapsulation of EDI documents in mime format for transfer via email or ftp is defined in rfc1767 (Crocker, 1995). For transmission using HTTP no standard is defined yet. With our approach workflow systems can be designed to react to receiving EDIFACT documents by starting or continuing a workflow process. Workflows then can result in the sending of EDIFACT documents to other sites. In this way we achieve a high potential for cooperative information systems while keeping the coupling between different systems at the low data coupling level.

References

Berge, J., editor (1994). *The Edifact Standards*. NCC Blackwell, 2nd edition.

Chakravarthy, S., editor (1992). *Data Engineering Bulletin, Special Issue on Active Databases*, volume 15.

Crocker, D. (1995). Rfc1767 - Mime encapsulation of EDI objects. InterNIC.

Dayal, U. (1988). Active database management systems. *Proc. of the Third International Conference on Data and Knowledge Engineering, Jerusalem.*

Eder, J., Groiss, H., and Nekvasil, H. (1994). A workflow system based on active databases. In Chroust, G., editor, *Connectivity 94*, pages 249–265. Oldenburg Verlag.

Eder, J., Kappel, G., and Werthner, H. (1986). Evaluation of scheduling mechanisms of a dynamic data model by simulation. In *Prof. of Int. Conference on Measurement and Control*, pages 86–91.

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

Hämmäinen, H., Eloranta, E., and Alasuvanto, J. (1990). Distributed form management. *ACM Transactions on Information Systems*, 8(1):50–76.

IBM (1994). *FlowMark: Managing Your Workflow*. IBM, Document No. SH19-8176-01.

Mohan, C., Alonso, G., Günthör, R., and Kamath, M. (1995). Exotica: A research perspective on workflow management systems. *Bulletin of the Technical Committee on Data Engineering*, 18(1):19–26.

Workflow Management Coalition (1994). Glossary - a workflow management coalition specification. Brussels, Belgium.

Yourdon, E. and Constantine, L. (1979). *Structured Design*. Prentice-Hall.

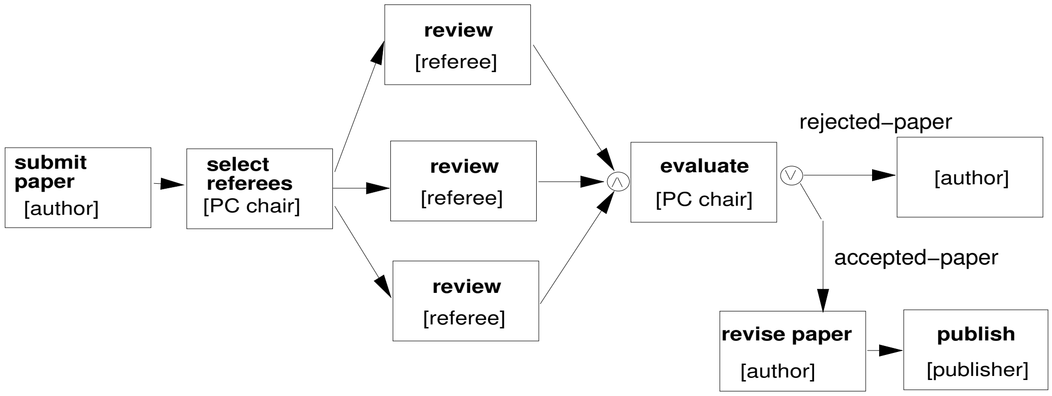
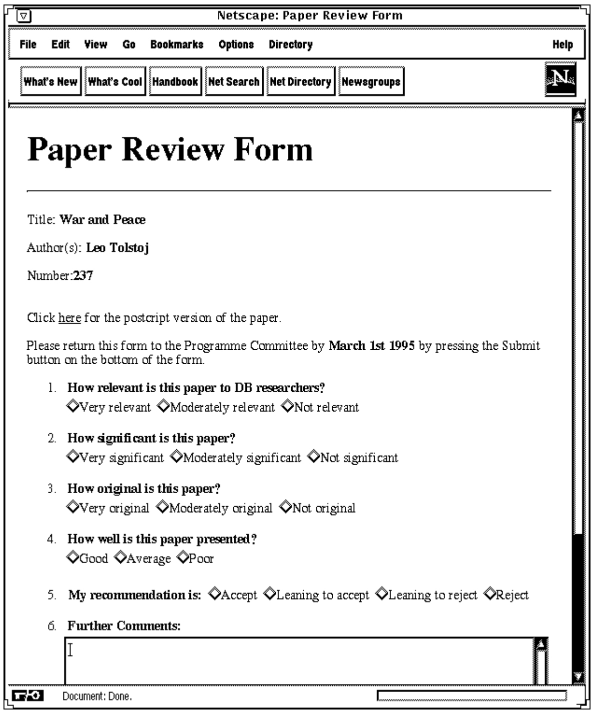


Fig. 1: WDL process diagram for a paper review



Paper Review Form

Title: **War and Peace**

Author(s): **Leo Tolstoj**

Number:**237**

Click here for the postcript version of the paper.

Please return this form to the Programme Committee by **March 1st 1995** by pressing the Submit button on the bottom of the form.

1. **How relevant is this paper to DB researchers?**
   ◇Very relevant ◇Moderately relevant ◇Not relevant
2. **How significant is this paper?**
   ◇Very significant ◇Moderately significant ◇Not significant
3. **How original is this paper?**
   ◇Very original ◇Moderately original ◇Not original
4. **How well is this paper presented?**
   ◇Good ◇Average ◇Poor
5. **My recommendation is:** ◇Accept ◇Leaning to accept ◇Leaning to reject ◇Reject
6. **Further Comments:**

Fig. 2: Paper review form in Netscape client

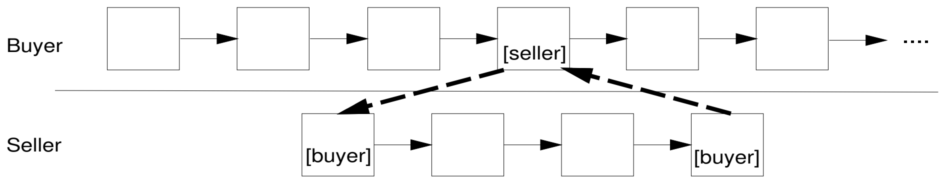


Fig. 3: Interaction of two workflows