A Workflow System Based on Active Databases

Johann Eder, Herbert Groiss, Harald Nekvasil

Institut für Informatik
Universität Klagenfurt
Universitätsstr. 65
A-9022 Klagenfurt, AUSTRIA
e-mail: {eder,herb,nekvasil}@ifi.uni-klu.ac.at

Abstract

We present the architecture of a workflow system based on active databases. Business processes can be defined in an easy-to-use graphical workflow description language. Process descriptions written in this language are compiled to rules and executed in a system based on an active database. Thus the workflow system can take full advantage of the capabilities of a database system such as reliability, recovery, concurrency control, transactions, and authorization. We describe the general architecture of such a system and the implementation of a first prototype and discuss the advantages of this approach for building as well as applying workflow machines.

Keywords: workflow management, active database, trigger, dynamic modeling

1 Introduction

The aim of workflow systems is to support business processes. From an abstract perspective a business process consists of a sequence of tasks. The process specifies

- which tasks have to be performed
- in which sequence (probably depending on decisions which are part of the process),
- by whom
- under which constraints (time, quality)

Such business processes can be found in businesses, industries, and administration. The tasks can be performed automatically, by humans or by interaction of humans with information technology (IT). Traditionally, business processes are mainly managed using paper, forms, and other communication media. Traditional IT supports business processes only in a rather limited way. It is restricted to standard processes and is conceived as very inflexible. But current economic changes flashlighted by buzzwords such as lean management, just in time production, and computer integrated manufacturing, require enterprises as well

as administrations to be highly reactive to external and internal events, to participate in tightly integrated processes and to be able to flexibly adjust these processes.
Advantages of applying workflow systems to business processes comprise the following:

- *Specification:* The application of workflow systems leads to a better specification of business processes, of regular (standard) processes and even more of special ad-hoc processes. Even if this is not a technical matter, experience shows that the organizational analysis and design needed to employ workflow systems increases the quality of business processes.

- *Documentation:* The application of workflow systems leads directly to an exact documentation of business processes. It should be noted that process documentation is an inherent necessary feature for quality management. This integrated documentation also yields better traceability of processes, built-in status accounting, and improved responsiveness.

- *Turn-around:* A primary goal for employing workflow systems is to reduce turn-around times and therefore to improve reactiveness.

- *Flexibility:* In comparison to traditional software solutions, workflow systems are much easier to adapt. They allow a very dynamic and flexible redesign of business processes to adapt to business needs. Furthermore, standard cases / processes as well as non-standard ones can be dealt within the range of one system.

- *Integration:* Workflow systems can act as 'glue' between different ITs allowing also the integration of legacy systems in new business processes.

The aim of our work is the automation of such business processes, also called workflows. This requires in a first step the storage of the documents handled by the agents in a database, and electronic forwarding of the documents from one agent to the next.
This is the conventional way using tools like text-processors, spreadsheets, databases and electronic mail. When only these tools are used the knowledge about and the responsibility for the process remains with the agents, who process the documents and decide then to which successor these documents have to be delivered.
An automation of this task requires:

a) a model (schema) of the process,

b) an automatic delivery mechanism for documents according to the process information.

c) a mechanism for automatic invocation of programs

We call the latter two items a workflow machine. This workflow machine is data and event driven and uses the process information to decide about the delivery of a document finished by an agent and the invocation of automatic agents.
In the last years systems for automating business processes have been studied in the area of office automation or office information systems [HEA90], [ML91], [TLF88] [EKTW87],

[MS93],[CB82],[LCS82],[Zlo82] . Only recently the term *workflow* was coined for such types of systems and interest in such systems exploded. More than 40 workflow management systems with quite different capabilities, are on the market today and most of them went to the marked in the past two years. There seem to be commercial as well as technological reasons for this rush. The commercial reasons for stimulating the demand for workflow management systems have been outlined above. The technological reasons are seen in the high availability of fast communication infrastructures, client server solutions, powerful client workstation, and the need to integrate legacy information systems.
The main 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 [Day88], [Cha92]). 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 a 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 rule are applied. Conditions are descriptions of database states, actions are operations, which can modify the database or start external procedures. In this paper we use the syntax of SQL3 [IA93], where the basic structure of a rule is:

`create trigger` *name* `on` *table*
`after` *event*
`when` *condition*
`then` *action*

With create trigger a rule is defined, which reacts on changes of the table *table*. The *event*, which triggers the rule is specified next and the *conditions* - a SQL query - follow the keyword `when`. Actions are database actions formulated in SQL.
Because the description of the processes in terms of triggers is on a very low level, such programs are hard to read and to debug. Therefore, we describe the workflows in an easy-to-use graphical high level language designed specifically for this purpose and translate the specifications of workflows into triggers of an active database system. This has also the advantage of independence of the descriptions from a specific product or trigger language.

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 reimplement 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 [EKW86].

In the next section we introduce the language designed for specifying the processes, in section 3 the translation process of a workflow description to the rules of the active database and in section 4 the system architecture of our prototype implementation is described.

2 The workflow description language - WDL

The main design criteria of WDL were: easy to use for an end user to design simple workflows, flexibility in describing a wide variety of business processes, direct compilation to an executable workflow. Note that WDL describes only the communication between tasks, i.e. the data flow and control structure between tasks, but does not specity the internal structure of a task or which modifications a task performs.
The basic modelling concepts are:

- users and roles,
- forms, and
- processes, consisting of tasks and flows.

At first a brief description of these concepts is given:

user: describes an agent, who can perform tasks (some data manipulation)

role: defines a set of users with common properties (e.g. clerk) or as members of an organizational unit

form: data container holding the information that flows between the different agents. Forms are used for representing and manipulation of information

process: describes the structure of a complex, distributed job; i.e. which tasks and flows it is composed of

task: defines an elementary activity (i.e., done by one agent)

flow: defines the transmission of information (a form) between two agents

workflow: aggregation of processes

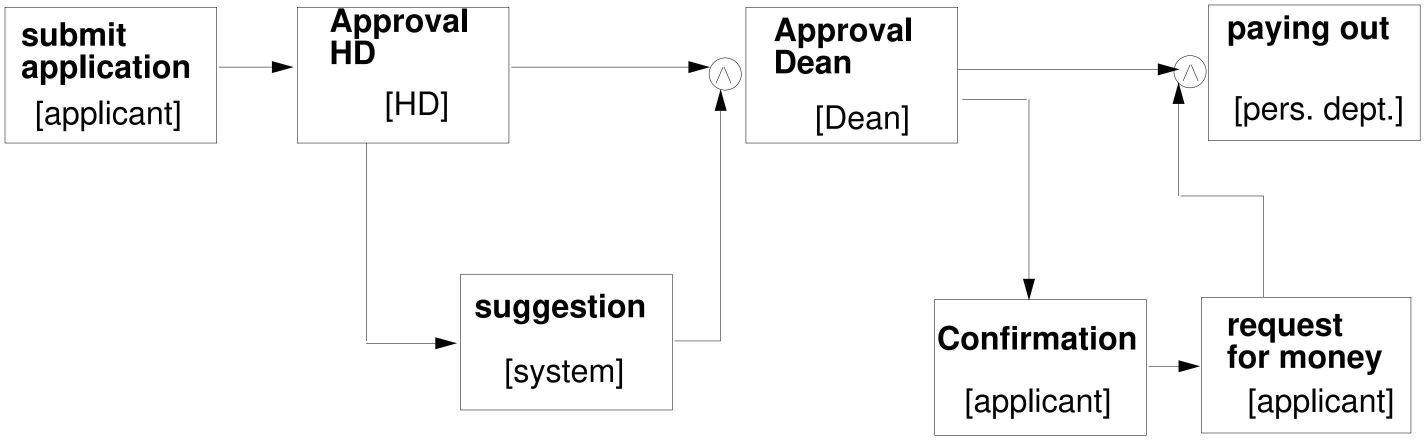


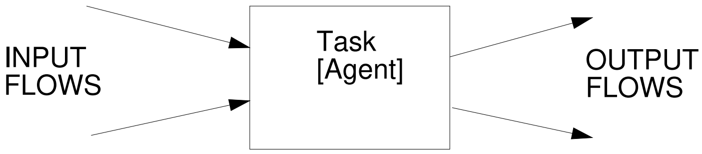
Figure 1 WDL process diagram for a business trip

Many concepts of the modelling language can be expressed in a comprehensive WDL process diagram.

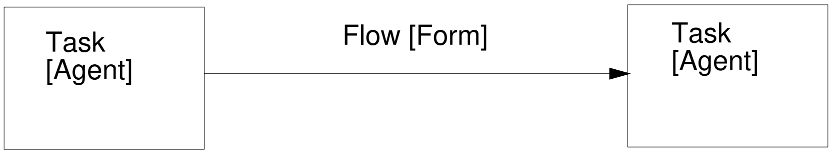
2.1 Description of the graphical notation

A process diagram specifies the structure of one specific process involving the tasks, the data flows between them and users respectively the roles performing these tasks.
Fig. 1 shows such a diagram for the process of an application for a business trip. This process requires interaction of different persons and departments. An applicant who plans to make a trip needs a permission from the head of the department and the dean. After the trip he gets the money from the personal department.
The main elements of the graphic representation are tasks and flows:
A task is an elementary activity done by one person or one computer program. What exactly happens when a task is executed is not in the focus of the description, typically a task changes the contents of some forms.
A task is represented by an rectangle. 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 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. It is also possible to specify a task

timeout. That means after the specified duration a timeout is signalled.

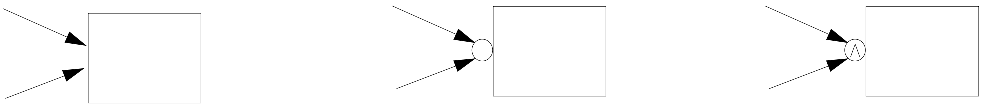


A flow connects two tasks and is denoted by an arrow. Considering one task you can group the flows into input flows and output flows. Input flows are the incoming flows of a task delivering the necessary forms to the task. Output flows specify the outgoing forms after completion of the task. The delay for a flow can be defined similar to tasks. The form is then transmitted after the specified delay.



Usually a task will have more than one input and output flow. Therefore we introduce the following concepts for specifying constraints on the input and output side of a task:

1. Input side:



We can define some preconditions which must be fulfilled before activating the task. Possible preconditions are:

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], that 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 modelling parallel manipulation a form 'splitting' at the begin and a synchronization point at the end has to be defined.

Note that the concept of dynamic users is very powerful, for example e-mail can be modelled 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.

2.2 Extensions to the graphical notation

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 you have 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:

   a) postconditions:
   You can explicitly define some postconditions to enforce a valid state. The task can only be successful completed 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.
In section 4 we describe a graphical design tool facilitating the specification of workflows.

2.3 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 Translating a workflow description into rules

In this section we describe the principles of the implementation of a workflow system based on WDL with active databases. The whole description of a process in WDL is stored in the rules and tables of the database.

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 (called flow-rule), 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.

The following types of rules are generated for each task:

post-task rule: This rule triggers when the task is finished and executes the postprocedure (step 1 of the execution model).

postcondition rule: The rule tests the postconditions of the task (step 2).

precondition rule: This rule tests the precondition of a tasks: This is necessary if the task has more than one input flow. On each arrival of a form at the task this rule is triggered and checks whether all forms necessary for the execution of the task are available (step 4).

dispatch rule: The rule exists, whenever the performer of the task is specified as a role together with a selection criterion (step 5). The non-empty user field of the task after the execution triggers the next rule:

pre-task rule: This rule applies the before procedure. After completion this rule sends a signal to the client program (either the standard client or an application program performing the task).

We want to emphasize that the whole workflow manager 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.

4 System Architecture

To evaluate our approach we implemented a prototype workflow manager using the ORACLE database management system version 7, which provides a simple rule system. As hardware platform we use a cluster of SUN workstations with SUNOS 4.1.2 and OpenWindows.
The system consists of four components:

- the server,
- the user interface client,
- the workflow design tool and the WDL compiler,
- the monitoring client.

The server is the active database management system with the rules, forms, tasks and users specified in the database. No additional code is necessary, because the communication of the other components is done exclusively via database accesses.

4.1 user client

This component is the interface of the normal user to the workflow system, Fig. 2 shows the appearance on the screen. The typical process of handling is similar to the processing of mail:
The user interface notifies the reception of a task. When the user selects a task, he gets a task description with some general information (sender, corresponding process, description of the task, etc.) and a list of forms. He can now view and edit the received forms. In this step the user can only see the forms and fields which are marked as visible or editable in the task description and can only edit the fields declared as editable. During filling in the forms

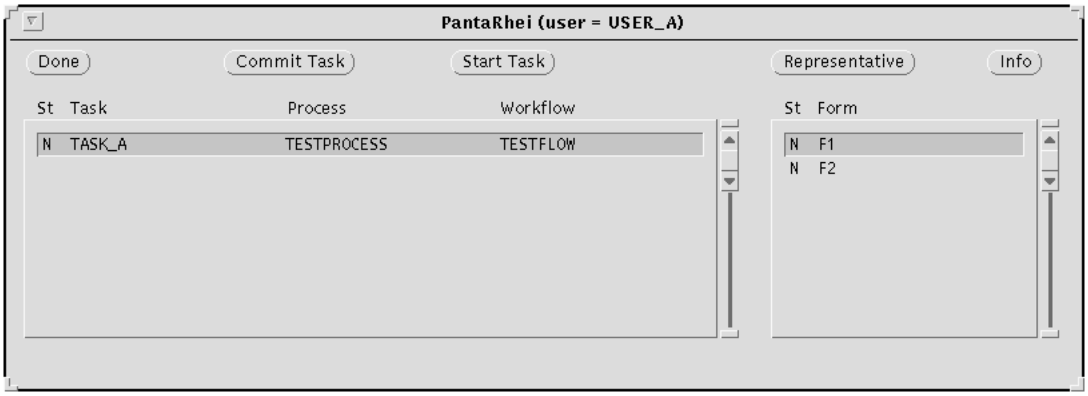


Figure 2 user client interface

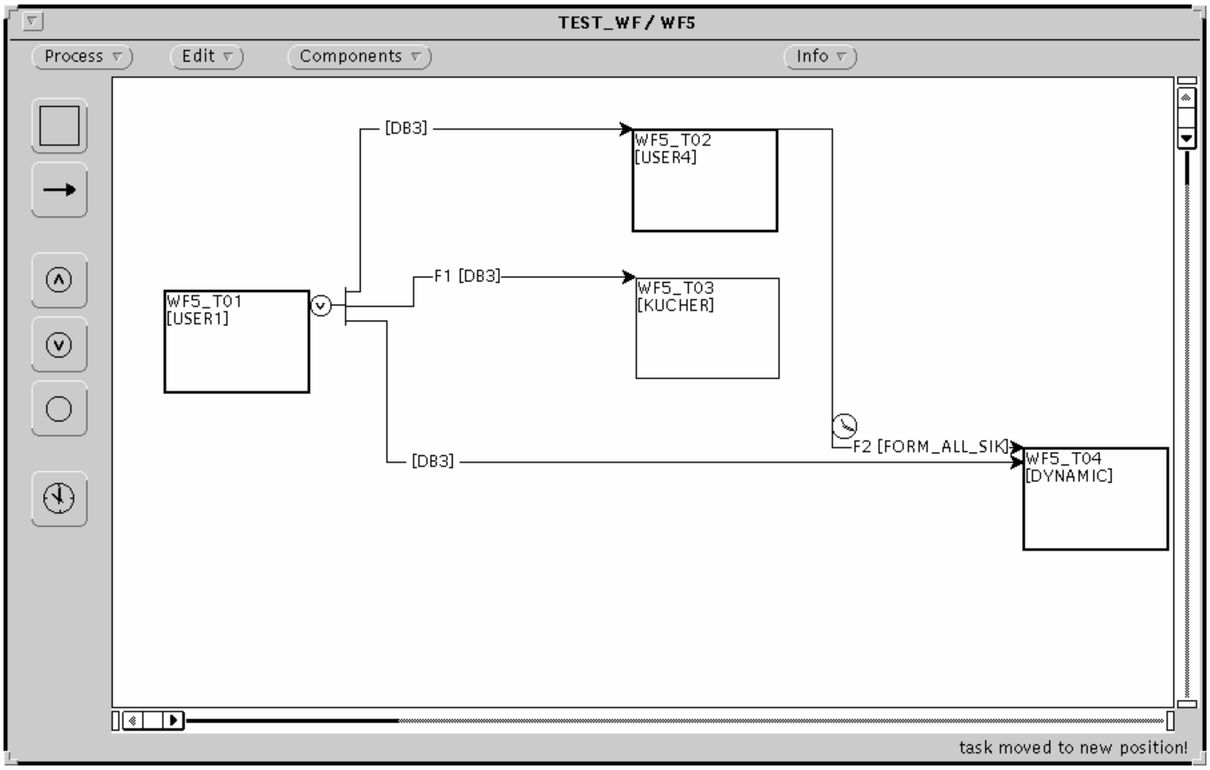


Figure 3 designer interface

the user can rollback the modifications of each form. The work with a task is concluded with a commit, which results in communication with the server for running the post-procedure, checking the post-condition, and removing the task from the users active-task list.
The explicit archivation of the forms is not necessary, as the history mechanism of the server keeps the whole history of each form. Every user can view all forms he handled in the past. Moreover the user can send copies of the forms to other users like ordinary mail. This allows informal communication in addition to predefined workflows.

4.2 workflow designer interface

The purpose of this component is to allow an interactive graphic design of workflow processes and forms. The user interface of the process designer is shown in Fig. 3. The second part of this tool is the compiler which translates the WDL workflow description into the rules of the active database. The maintenance of the processes and workflows is done by the database. The designer can use previously defined forms and tasks. After a newly defined process is stored in the database, it is possible for the users responsible for the initial task to initiate the process.

4.3 monitoring client

An important task in workflow systems is monitoring, e.g. inspecting which forms are pending, how long are the active task lists of the different users, etc.
The structure of this component is very similar to the ordinary user interface. The main difference is, that all forms currently in the system are visible. With different views all forms of a type or all forms belonging to a process can be viewed. The contents of the forms can be edited.
In addition, the monitoring client is used for maintenance tasks like installing users.
The implementation of this component was very simple due to the availability of all needed data in the database. The information about the stati of the tasks, the location of forms or the workload of users can be retrieved with simple SQL-queries.

5 Conclusions

In this paper we proposed a new approach for the development of workflow management systems. We presented a workflow description language for the graphically assisted specification of workflow processes. The goals for the design of the language were to make it as easy as possible such that also (skilled) endusers may use it to define workflows in an ad-hoc manner and on the other hand that it scales up to be able to be used for all processes. To make the language simple we use well known metaphors like business forms and support the specification of workflows with a graphic workflow designer tool. Since the language supports the specification of arbitrary conditions and features higher order constructs such as the specification of a receiver as part of a task, it scales up to represent workflows of any complexity.

The most important contribution of our approach is to show how active databases can be used to facilitate the development of workflow management systems and the application of such systems. Modern database management system are capable of storing and manipulating any kind of data, so it is quite natural to use database systems to maintain all data relevant for business processes. The advantages of our approach can be summarized as follows:

efficiency: This approach showed to be very efficient for the development of a workflow management system since it can use directly all the features of a database management system like transaction management, concurrency control, access authorization and recovery. So the necessary code for a workflow management system can be minimized. This approach is also very efficient for the actual processing of workflows, since the trigger concept of active databases is a very efficient way to schedule tasks, transport data between tasks and launch processes.

reliability: All relevant dynamic information about processes is mapped into the database. So the recovery mechanism of the database management system is used for storing the data as well as the state information of all workflow processes in a reliable way. In the case of system failures not only the data but also all the information about processes are recovered. A further aspect of this approach is that no user or program can circumvent the workflow manager - be it intensionally or by accident. All changes to data relevant for a workflow are monitored by the active database and, therefore, by the workflow manager.

extendability: The workflow description language allows an easy extension of workflows. Furthermore, since all changes to data are monitored by the workflow manager, arbitrary existing application programs can be used within workflows without changing them. They can be automatically launched from the workflow manager and the changes they perform on data can immediately trigger workflow processes. So the tight integration of workflow manager and database system facilitates the development of workflows as integration platform for existing isolated applications.

traceability: Since all changes to relevant data are managed or monitored by the workflow system, all such changes can be automatically documented. All business processes under control of the workflow manager are documented and can be traced - meeting an important requirement of quality assurance procedures without additional effort.

We have successfully applied this approach in the developement of a prototype workflow management system. Active databases have proven to be a powerful technology for implementing such a system. The software engineering problems arising in programming with rules have been avoided through the usage of a higher level language for describing business processes. The usage of a standard commercial database brought the benefits of a stable, system available on different platforms, but had the drawbacks of a limited trigger mechanism: In Oracle it is not possible to use triggers for changing the table which initiated the trigger application. Moreover, triggers reacting on temporal events are not supported.

In the future we plan to extend our system in several directions. We will integrate an extended transaction concept, allowing long running activities accompanied with a compensation mechanism for handling exceptions (e.g. cancellations) requiring the description of inverse tasks and inverse activities. We will work on a characterization of well-formed processes and check processes for well-definedness: e.g. two parallel tasks should not alter the same attribute, or: reading a value requires earlier writing. Finally we will port the prototype system to other database management systems following the object-oriented or the object-relational paradigm.

Acknowledgements: The authors thank Werner Liegl, Jürgen Modre, and Michael Stark for their efforts implementing the workflow designer interface.

References

[CB82] C.A.Ellis and M. Bernal. Office-Talk-D: An Experimental Office Information System. In *Proc. First ACM-SIGOA Conference on Office Information Systems*, 1982.

[Cha92] S. Chakravarthy, editor. *Data Engineering Bulletin, Special Issue on Active Databases*, volume 15. December 1992.

[Day88] Umeshar Dayal. Active Database Management Systems. *Proc. of the Third International Conference on Data and Knowledge Engineering, Jerusalem*, 1988.

[EKTW87] J. Eder, G. Kappel, A. M. Tjoa, and R.R. Wagner. A Behaviour Design Methodology for Form Flow Systems. Technical report, Universität Klagenfurt, Institut für Informatik, 1987.

[EKW86] J. Eder, G. Kappel, and H. Werthner. Evaluation of Scheduling Mechanisms of a Dynamic Data Model by Simulation. In *Prof. of Int. Conference on Measurement and Control*, pages 86–91, 1986.

[GE93] H. Groiss and J. Eder. Active Databases and CIM (in German). In *Tagungsband des ADV-Kongresses Informations- und Kommunikationstechnologie für das neue Europa*, pages 861–870, 1993.

[HEA90] Heikki Hämmäinen, Eero Eloranta, and Jari Alasuvanto. Distributed Form Management. *ACM Transactions on Information Systems*, 8(1):50–76, January 1990.

[IA93] ISO and ANSI. Working Draft Database Language SQL (SQL3). Digital Equipment Corp. Maynard, MA, August 1993.

[LCS82] V.Y. Lum, D.M. Choy, and N.C. Shu. OPAS: An Office Procedure Automation System. *IBM Systems Journal*, 21(3), 1982.

[ML91] Clarence Martens and Frederick H. Lochovsky. OASIS: A Programming Environment for Implementing Distributed Organizational Support Systems. *SIGOIS Bulletin*, 12(2):29–42, 1991.

[MS93] D. R. McCarthy and S. K. Sarim. Workflow and Transactions in InConcert. *Data Engineering Bulletin*, 16(2), June 1993.

[TLF88] Michel Tueni, Jianzhong Li, and Pascal Fares. AMS: A Knowledge-based Approach to Task Representation, Organization and Coordiantion. *SIGOIS Bulletin*, 9(2):78–87, April 1988.

[Zlo82] M.M. Zloof. Office-By-Example: A Business Language that Unifies Data and Word Processing and Electronic Mail. *IBM Systems Journal*, 21(3), 1982.