WORKFLOW MANAGEMENT AND DATABASES

Johann Eder, Herbert Groiss, Walter Liebhart

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

ABSTRACT

Workflow management systems are among the most interesting concepts for supporting modern organizations with a focus on processes rather than on structure. Workflow management systems offer different degrees of automation of business processes. We classify workflow management systems according to the features they provide and the types of processes they support. Database systems facilitate the realization of workflow management systems in several ways. They can provide the necessary functionality to keep the workflow relevant data, business data as well as process data. The dynamic execution of workflows can be handled by triggers of active database systems. Furthermore, the transaction concept can be extended to develop workflow transactions for consistent execution of workflows and intelligent treatment of exceptions and errors.

1 Introduction

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], [EB82], [LCS82], [Zlo82] . Only recently the term *workflow* was coined for such types of systems and interest in such systems exploded. More than 60 workflow management systems with quite different capabilities are on the market today and most of them went to the market in the past four 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 will be outlined below. 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 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 flash-lighted 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.

In this paper we will give an overview of the architecture of workflow management systems. We will show that different targets led to the development of different types of systems and will discuss a classification of technical aspects of workflow management systems which influence their area of application. In particular, we will study how database technology contributes to the development of workflow systems and will emphasize two aspects in particular. We present the architecture of our prototype workflow management system Panta Rhei, which uses databases not only as a tool for the management of workflow relevant data but maps the dynamic workflow engine to rules of an active database system. The second aspect is the development of workflow transactions by extending database transaction concepts as means for permitting consistent concurrent execution of workflows and as a basis for intelligent automatic treatment of exceptions and errors during workflow execution.

2 Workflow and Workflow-Management

The workflow concept has evolved from the notion of *process* in manufacturing and the office. A process is often defined as a set of partial ordered steps with the intention to reach a special goal. Processes typically consist of *process elements* which can be further decomposed until atomic process elements, *process steps*, are reached. The terms workflow and process are often used synonymously. In this paper we also will not distinguish between process and workflow. Additionally, we use the term *activity* for (complex) process elements and the term *task* for (atomic) process steps. For the rest of this paper we do only distinguish between activities and tasks if it is really necessary. If we use the term activity then it should be clear from the context whether a complex activity or an elementary task is meant.

Workflow management involves the coordinated execution of a business process, consisting of several activities and tasks which are performed (partially) automatic by an information system or manually by a human. WFMSs offer an environment to define and execute such processes. According to the *Workflow Management Coalition* (WMC), a WFMS is a system that completely defines, manages and executes business processes through the execution of software whose order of execution is driven by a computer representation of the process logic [Wor94]. Based on this definition we can identify the following main topics of workflow management:

- *Workflow specification:* requires workflow models and methodologies for capturing a process as a workflow specification.

- *Workflow implementation and execution:* requires methodologies/technology for using information systems, and human performers to implement, schedule, execute, and control the workflow tasks as described by the workflow specification.

The architecture of a WFMS can best be illustrated by the workflow reference model defined by the WFMC [Wor94]. The model, as presented in Fig. 1 has been developed from the generic workflow application structure by identifying the interfaces within this structure which enable products to interoperate at a variety of levels. The interface around the workflow enactment service is designated WAPI (Workflow APplication Interfaces and interchange formats), which can be considered as a set of constructs by which the service of the workflow system may be accessed and which regulate the interaction between the workflow control software and other system components.

2.1 Workflow Classification

Up to now there exists no general accepted classification framework for workflows (processes) and workflow systems. Since every classifications focuses on some specific aspects, it will always be difficult and probably impossible to give a commonly accepted classification.

In [GHS95] workflows are characterized along a continuum from *human-oriented* to *system-oriented workflows*. In the first case, a workflow is mainly performed by human agents. The WFMS is expected to support the coordination and collaboration of humans who are responsible for consistent workflow results. In the second case, workflows are characterized as highly automated and computation-intensive processes which involve the integration of heterogeneous,

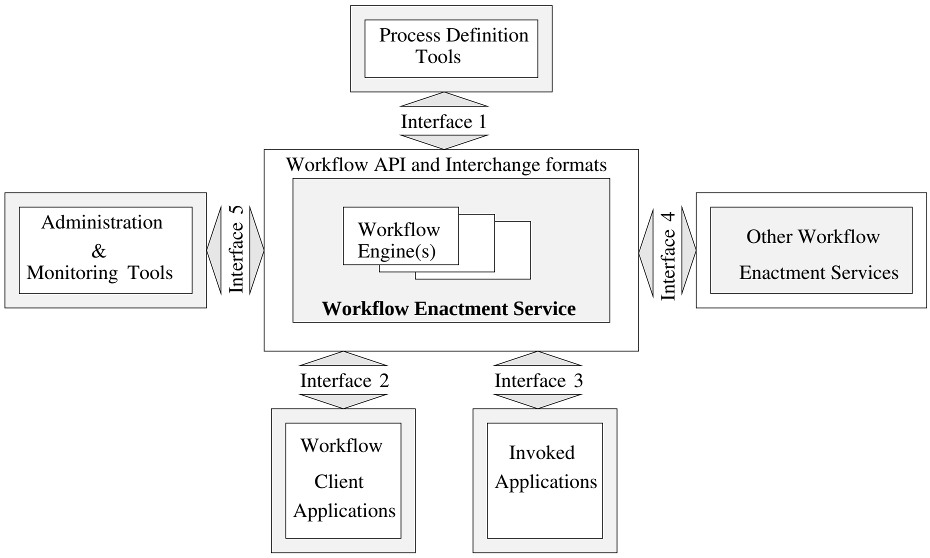

Fig. 1: Workflow reference model - components and interfaces

autonomous and / or distributed information systems. Since human influence is very limited, system-oriented workflows must include software for various concurrency control and recovery techniques to ensure consistency and reliability.

Trade press [McC92] often distinguishes between ad hoc, administrative, and production workflow. The main differences between these types comprise (1) repetitiveness and predictability of workflows and tasks, (2) how the workflow is controlled (e.g., human controlled or automated) and (3) requirements for WFMS functionality. *Ad hoc workflows* typically consist of several non automatable tasks which need the coordination of humans. Additionally, the ordering and coordination decisions are made *while* the workflow is performed. *Administrative workflows* describe more structured and predictable processes in organizations which involve manual and / or simple automated tasks. The main difference compared to ad hoc processes is that the coordination rules and dependencies between tasks in administrative workflows can be defined *before* the process is activated. In contrast to administrative workflows, *production workflows* typically represent complex business processes consisting of highly-automated tasks which are performed on heterogeneous, autonomous and / or distributed information systems. The processes are predictable, repetitive and, therefore, the execution order of the various tasks can be performed automatically with little or no human intervention (e.g. trip reservation, loan requests, insurance claims or telecommunication processes).

As we will see at the end of this subsection, our classification approach [EL96a] leads to similar results but it points out several workflow specific features which are helpful for transaction management requirements. The classification is based on two pillars: the static and the dynamic aspects of a workflow.

2.1.1 Static Aspects of a Workflow

The static aspects of a workflow comprise all components which can be extracted from a workflow meta-model. We use the meta-model presented in subsection 2.2. There, the basic elements of a workflow are *activities*, *data objects* and *agents*. Within our static classification part, these constructs are further characterized by the following properties:

- *Activities*: Activities are characterized by several attributes (like name or state) and methods (e.g., start, succeed). Elementary activities (tasks) describe the real work items in a process. We can distinguish between manual and automatic (either interactive or non interactive) tasks. *Manual tasks* are performed mainly by human agents. This includes in particular the manual start and manual termination of the task. The work which is performed within a manual task is fully under the control of the human agent (e.g. making a phone call, writing a letter, etc.). The WFMS only supports the agent by providing him with appropriate standard tools (like a text processor, etc.), if necessary. *Interactive automatic tasks* are associated with *specific* programs (software applications) which are executed after a responsible human agent has selected the task from his worklist. During task execution the agent communicates interactively with the associated program. As soon as the program terminates also the task reaches its termination state. *Non interactive automatic tasks (batch tasks)* are specific software programs which are started by the WFMS and which are fully executed under the control of the WFMS. The execution of the task needs no human interaction. The task terminates as soon as the associated (batch) program has finished.

- *Data objects*: Within a workflow different kinds of data are manipulated. Mainly, we have to distinguish between data which is manipulated within the tasks and data which is needed for process execution (scheduling, etc). These different kinds of data are not necessarily disjunctive (e.g., the amount of a loan in a loan request workflow is not only used within specific tasks but also necessary for the workflow scheduler in order to find the correct path in the process). More precisely, in a workflow application we find the following data types:

  - *Application data:* The application data is consumed and produced by the tasks (applications) in a workflow. Additionally, this kind of data can further be classified into *structured*, *unstructured* or *semi-structured* data. The main reason for this differentiation is to distinguish between data which can not only be used within tasks but also for process definition. Structured data (e.g. formatted data in a form) can be used easier in a workflow specification than unstructured data (like documents).
  - *Process data:* Process data are necessary to define and control the execution of workflows. Typical examples are the state of tasks, the start time of a task and so on.

- *Agents*: An important function of a WFMS is to assign tasks to agents (users or programs) who are eligible to carry them out. The modeling and definition of agents composes very simple but also very sophisticated approaches [BJ94]. For our classification it is only necessary to distinguish between human and machine agents.

2.1.2 Dynamic Aspects of a Workflow

The execution of a workflow mainly comprises the answer to the following question: *What (activity) has to be executed when, by whom and with which data?* In [Jab94] these W-questions are termed as functional (what), behavioral (when), organizational (whom) and informational (which data) perspectives. We want to emphasize that the behavioral aspect (temporal execution order of the various activities within a workflow) is a central topic. Based on this perception, we informally define a workflow as *the definition and/or execution of correct activity-sequences.* Of course, a central question now is, how correct activity-sequences can be defined or achieved. We have identified two possibilities:

- *Ad hoc and without a corresponding formalism:* Within this approach correct activity sequences are determined by human agents ad hoc during workflow execution (at run time). Additionally, we distinguish whether the correct sequences are defined with or without *predefined* activities. In the first case, a workflow is composed of already defined parts at run time while in the other case the agents have the possibility to define *new activities* (and hence workflows) during run-time.

  This concept is also valid for WFMSs which are not based on activities but on agents (e.g. email based WFMSs). In this terminology each agent has to decide who should be triggered next in order to perform some work (which corresponds to an activity).

- *With a corresponding formalism:* Valid activity sequences are defined during workflow modeling time by the workflow designer. Therefore WFMSs offer (more or less comfortable) modeling tools which allow the definition of business processes based on a special formalism. There exist different formalism types which are discussed in [EL96a]. Main differences between the various formalisms depend (1) on the available process information and (2) on the access to application data.

2.1.3 Workflow Types

Based on the previous classification features, we are now able to identify at least two main types of workflows:

- *Document-oriented workflows:* Document-oriented workflows are mainly characterized by the existence of manual tasks and unstructured or semi-structured documents. The execution of the workflows are primarily controlled by human agents. They normally have to decide during workflow execution *when* a specific task is performed, *which* task should be executed next (in case there is a set of potential successor tasks) and *who* should execute the next task. The requirements for WFMSs within this area are to support the coordination and collaboration of humans who are responsible for consistent execution results.

- *Process-oriented workflows:* Process-oriented workflows involve mainly automatic tasks and structured or semi-structured data objects. The processes may be very complex and therefore adequate formalisms to define such processes are necessary. Additionally, the tasks in general perform complex operations based on autonomous, distributed legacy and new systems. WFMSs supporting this type of business processes control and coordinate the execution of the tasks with little or no human intervention. Therefore, various concurrency and recovery mechanisms are necessary to guarantee a reliable and consistent execution.

2.2 Workflow Specification and Execution

In order to perform workflow specification a *workflow meta-model* is necessary. Such a model typically includes a set of concepts which are useful to describe processes, their process steps (activities and/or tasks), the dependencies among process steps, and the required agents that are eligible to perform the specified process steps. The specification usually is based on a *workflow specification language*. These languages use rules, constraints, and/or graphical constructs to describe the structure of complex workflows. A simplified workflow meta-model is presented in Fig. 2.

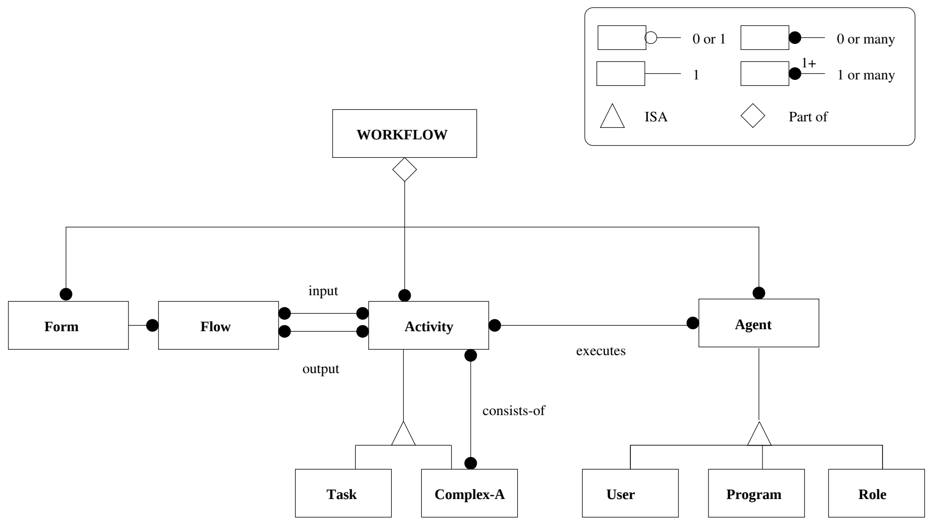

Fig. 2: A simple workflow meta-model

As already mentioned in subsection 2.1.2, during workflow specification the workflow designer is faced with the identification and modeling of functional, behavioral, organizational and informational aspects of a complex business process [CKO92, Jab94].

In the *functional part* of a workflow specification the workflow designer specifies *which* processes have to be performed. For this purpose, a complex workflow is decomposed into smaller sub-workflows (activities) until *elementary workflows* (tasks) are remaining. Workflows containing other workflows are often called *composite workflows* . Elementary workflows are associated with applications which implement the corresponding function. Applications may be any kind of executable code (e.g., programs, command procedures) but also some work which is fully performed by a human agent (e.g., to make a phone call). The information exchanged between the activities is stored in *forms*, the *flows* describe the path of these documents through the activities in which they are used. One important aspect in workflow modeling is that tasks are *black boxes* which are used and reused by the workflow designer. Only the interface (the signature) of a task needs to be visible in order to communicate with other tasks.

The execution of a hierarchical structured workflow starts at the top-level (most abstract) workflow in the hierarchy by executing the first underlying layer of sub-workflows. The execution order of the sub-workflows is determined by the behavioral information in the specification. Recursively processing continues until the bottommost level is reached where the referenced applications of the elementary workflows are executed. Before and after a single task is executed, several steps have to be performed.

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

2. The postconditions of the task are evaluated. If the postconditions are fulfilled, the documents 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 the task 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. A signal is sent to the program or the user client performing the task.

In the *behavioral part* of a workflow specification it is defined *when* processes are performed. The inter-dependencies between complex and elementary workflows are determined through the *control flow*. Of course, the specification of the control flow is not always possible and wished at modeling time, as for example in ad-hoc workflows. The most important control flow concepts are *serial* execution, *alternative* execution and *parallel* execution.

Within the *organizational part* of a workflow specification the workflow designer specifies *who* is intended to execute a workflow (or the corresponding application of an elementary workflow). To achieve a higher degree of flexibility often a simple *role-concept* is used. This means that the executor of a process is not directly connected to a user but to a role (e.g. a clerk, a technical assistant) which is an abstract description of certain skills which are necessary to perform a task. Roles are further associated with one or more users or to an information system. At runtime the system selects one of the agents who is defined by the role to execute the task.

The *informational part* of a workflow specification deals with the *data flow* aspects and the definition of information elements which are manipulated by workflows. As already discussed in subsection 2.1.1, workflow applications are faced with application data and process data.

3 Database-based Workflow Systems

There are two different approaches in implementing workflow management systems: the so-called *mail-based* systems exchange documents between the different agents via email or a similar mechanism. The *database-based* systems store the documents and the process information in a database to which all agents (users and programs) must have access to. This approach has several advantages:

- The execution of a workflow is a typical task for a client-server application. Normally, many clients from different location, probably running on different platforms connect to the workflow server. Crashes of one of these systems or disconnections in the network are therefore very likely. The recovery mechanism of the database management system ensures that after crashes of clients or the server a consistent state of every running process is restored.

- Central storage of all documents in the database ensures document integrity, preventing that different users work on different copies of a document, an explicit versioning can be handled by the database.

- Workflow processes should provide a high degree of concurrent execution to decrease turnaround times. The transaction mechanism permits to increase concurrency in a safe way. For example, it is possible that different users view a document concurrently, or different users edit different parts of the same document. 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.

- Modern database management systems provide application programming interfaces (APIs) to various languages and allows access over the network. This features are necessary, when coupling applications with the workflow system. The applications can communicate with the server only by making selects and updates in the server's database.

- The presence of all information about the dynamic state of processes and tasks in the database allows an easy implementation of a monitoring component. This part of the workflow management system allows the users and system operators to inspect the state of the processes, the content of documents, the work-lists of individual users, etc. All this can be retrieved by simple SQL queries. The authorization system of the database is used to control the different privileges of the users.

In very large systems the single central server architecture can be a performance problem, in this case is is possible to use a net of distributed databases, which share the information about the processes. Furthermore, the execution of a business process can be seen as a long transaction, needing more than the traditional built-in transaction mechanisms (concurrency control and recovery) of the database management system. This topic is discussed in section 5.

4 Using Active Databases for Workflow Execution

In this section we show how active databases can be used for implementing the workflow server.

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. (see Fig. 3). The specification of Event, Condition and Actions is done declaratively with so-called ECA-rules.

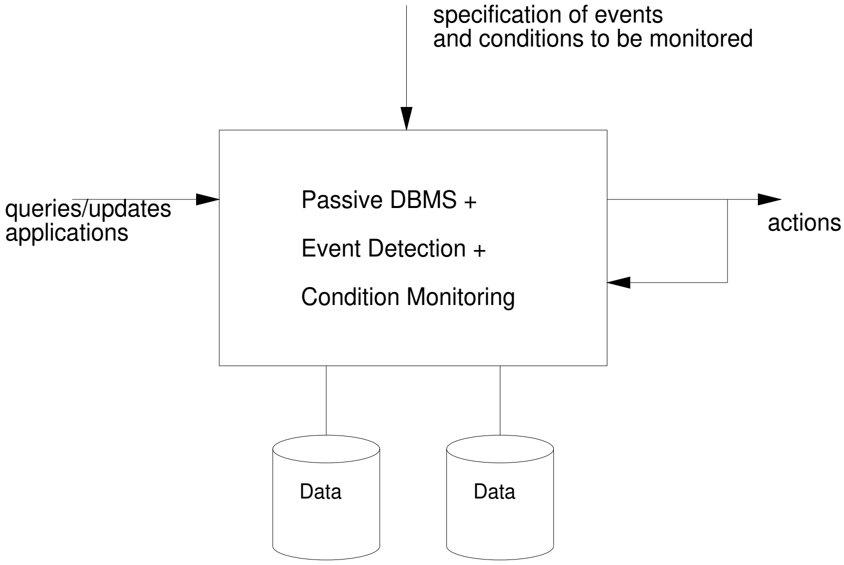

Fig. 3: Principle of an active database

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 the 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 [IA95], 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.

4.1 Translating a Workflow Description into Rules

The description of a process 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 also maintained in the database. 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 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_i 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 after-procedure (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 workflow specifications. All other necessary features are already provided by the database management system.

4.2 Advantages of Active Databases

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.
- 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].
- The development of the workflow server is very easy because only the rule compiler has to be implemented, the trigger mechanism is already part of the database.
- Once all dynamic process information is stored in the database, monitoring the processes can be done with queries to the databases and components based upon such queries. No additional bookkeeping is necessary.
- Fields in the data forms can be easily used for control purposes: For example, a form field can be used to specify the performer of the next task. The database trigger assigning the user, which is generated from the process description simply read the value of the field and assigns the task to this value.
- Existing applications can trigger a workflow: It is possible to define rules, that react on changes performed by other applications. For example an application, which handles the selling process would modify the store information. A rule can be defined which triggers when some amounts in the store fall below a threshold and starts an order process.

Using conventional techniques the implementation of a workflow management system is possible in two ways, each having some drawbacks:

1. A special application program polls the database, registers changes and reacts appropriate. This application can be a conventional program or a rule interpreter coupled to the database.

   This approach is insufficient in several respects: If the poll rate is too low, the reaction time could be too low for some application, because database state changes are recognized too late or even get lost. Is the poll rate too high, computing power is wasted continuously.

2. The second possibility is to enhance the application programs which perform the data changes, to inform the workflow server or make the necessary actions.

   This approach is against software engineering principles: The same functionality is distributed to different programs, each change on these features would make the change of all the applications necessary, resulting in inacceptable maintenance costs. Additionally, the integration of existing applications is impeded.

4.3 The System Panta Rhei

To evaluate our approach we implemented a prototype workflow system [EGN94] 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 5.4 and OpenWindows.

The system consists of five components:

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

The *server* is the active database management system with the rules, forms, tasks and users specified in the database. Additional code is only necessary for informing the client programs from changes in the database. For this purpose a process communicates with the database via a pipe and is awaked when it receives a message.

This *user interface client* is the tool for performing tasks by the user. The user interface notifies the reception of a task. When the user selects a task in his active-task list, 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 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 handled in the past.

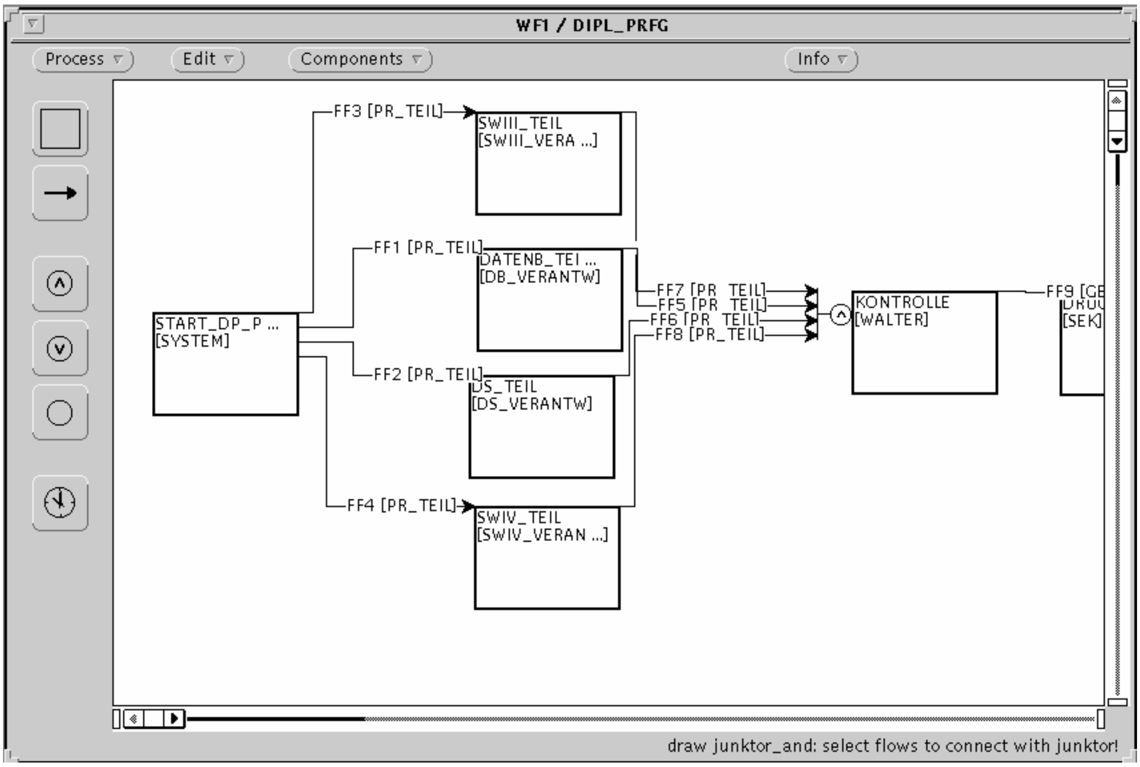

Fig. 4: Designer interface

Moreover the user can send copies of the forms to other users like ordinary mail. This allows informal communication in addition to predefined workflows.

The *workflow designer interface* allows an interactive graphic design of workflow processes and forms. Fig. 4 shows the appearance on the screen. The maintenance of the processes and workflows is done in the database. The designer can use previously defined forms and tasks.

Processes designed with this tool can be compiled to executable descriptions with the *workflow description compiler*. After a newly defined process is stored in the database and compiled, it is possible for the users responsible for the initial task to initiate the process.

An important part of workflow systems is the *monitoring component* which allows the users to inspect information about running processes, e.g. 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 tasks and documents currently in the system are visible. With different views all documents of a type or all documents belonging to a process can be viewed. The contents of the documents 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 state of the tasks, the location of forms or the workload of users can be retrieved with simple SQL-queries.

5 Transaction Support for Workflow Management Systems

Workflow applications typically are embedded into a *multi-user and non failure-free* environment. To guarantee correct and reliable execution of a complex workflow under such circumstances the WFMS must support the enforcement of consistency, concurrency control and recovery from failure. As similar problems are well known and more or less solved in the database area it is quite normal and reasonable to reuse these concepts and adapt them for workflow applications. The aim is, to use the concept of *transactions* in order to describe the correctness and reliability requirements of individual workflow applications.

5.1 Workflow Correctness and Reliability

Workflow correctness mainly deals with the correct and consistent execution of workflows. The following correctness aspects are relevant:

- *Consistency of individual tasks:* Usually, the person who implements a task is responsible for the correctness of the task, under the assumption that the task is executed alone and not interleaved with other tasks.
- *Consistency of individual workflows* (especially the concurrent execution of tasks that belong to the same workflow): It is reasonable to assume that if correct tasks are executed one after another in a previously specified order (defined during workflow specification), then the whole (workflow) execution preserves consistency. However, if multiple tasks of *one and the same workflow* are executed in parallel and they share common resources, their individual operations may interleave in such a way that incorrect results ((data) consistency problems) are produced.
- *Consistency between concurrent executions of different workflows:* Additionally, the problem of concurrency becomes much more problematic if tasks of *different* workflow types (or even other applications) access common resources.

Workflow reliability comprises restoring consistency if any kind of failure occurs. Potential *failure sources* in a workflow are (1) workflow engine failures, (2) activity (task) failures and (3) communication failures (see figure 5).

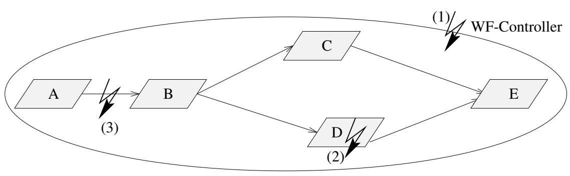

Fig. 5: Basic failure sources

A failure within the workflow engine (especially the workflow controller), like a system breakdown, in general demands a system restart. After a successful restart, *crash recovery* (often termed as *forward recovery*) must be guaranteed in the sense, that process execution is resumed from the point where the failure occurred. Failures within an activity are either system failures or semantic failures [EL96b]. A system failure (e.g. system breakdown, program failure) causes a non regular, abnormal termination (abort) of one or more *active* activities. A semantic or logical failure occurs, if an *activity fails* (commits unsuccessfully). This means, that the activity has terminated regularly but with a negative result (e.g., a flight cannot be booked because it is already over-booked). Communication failures comprise primarily the communication between workflow controller (workflow engine) and the involved activities (client and server communication). The coordinated execution of distributed activities demands appropriate communication protocols. This topic is not new and can be handled with adequate middleware-components, like TP-monitors [GR93] and distributed programming concepts (e.g. OMG CORBA [Gro92]).

Summing up, the main goal of workflow reliability is to restore - automatically if possible - the most recent consistent process state after a logical or system failure so that as few as possible work performed over long-durations is lost and process execution can be continued. Therefore *advanced recovery mechanisms* are required.

5.2 Workflow Transactions

In the database area *flat transactions* represent the most common (and simplest) type of transactions, and for almost all existing systems it is the only one that is supported at the application programming level. Nevertheless, they are too restrictive and inflexible for non-traditional applications, especially workflow applications. In general, they have the following characteristics:

- *Long duration:* Traditional transactions were invented for very short transactions whereas workflow activities have a much longer duration, touch many objects and have a complex control flow. Executing a long-running activity as a single ACID-transaction can significantly delay the execution of other high-priority short transactions. Additionally, it is not tolerable to rollback the whole workflow, and maybe the work of a day, if somewhere a failure (or exception) occurs. What is needed is application dependent (user-defined) failure atomicity.

- *Cooperation and concurrency*: In contrast to traditional applications, workflow activities are more of a cooperative nature where different sub-activities are allowed to concurrently access shared, persistent data (e.g. working on a common document). Of course, there is some kind of synchronization necessary to control the concurrent access but workflow applications in general have much weaker synchronization requirements than traditional applications - for example, they tolerate inconsistent results to some extend. Serializability as a global correctness criterion is not applicable in the workflow domain. Additionally, since workflow applications are of long-duration in most cases it is necessary to externalize uncommitted results or make them visible to other activities in order to achieve acceptable performance. Of course, since the results are uncommitted they may become invalid later in the process. This fact in general can be tolerated from an application point of view but it requires adequate mechanisms to handle such situations (e.g. backward recovery with compensation).

One of the most fundamental drawbacks of traditional database transaction in the context of long-running activity management is the fact that transactions are seen as concurrent and completely unrelated units of work. This means that there are no application independent system services for specifying inter-transaction dependencies - like control flow or semantic connections - except for putting all this control features into the application code. Additionally, flat transactions have only one layer of control which can be used by the application. Everything between `begin work` and `commit work` is at the same level which means that there is no way of committing or aborting parts of transactions, or committing results in several steps, and so forth. But especially transaction based workflow applications require not only an application independent communication service but also more dimensions of control in order to manage the control flow over distributed and autonomous applications constituting a business process. The ACID-properties as defined for flat transactions are in many aspects too rigid for workflow management.

These facts led to the development of more sophisticated transaction models, above all extended and relaxed transaction models, as for example summarized in [Elm92]. *Extended transactions* permit grouping of their operations into hierarchical structures (e.g. nested transactions) and *relaxed transactions* indicate that (some of) the ACID requirements are relaxed (e.g. open nested transactions relax the isolation requirement) [RS94]. However, in defining new transaction models it must be kept in mind that the success of flat transactions was its *simplicity* in isolating the application from faults. Extending transactions for non-traditional applications is not a case of "the more the better"; rather, a delicate balance between expressive power and usability (simplicity) [GR93].

As stated in [AKA+95a, RS94] most of the advanced transaction models for non-traditional applications are developed from a database point of view, where preserving the consistency of the shared database by using transactions is the main objective. A basic fact behind these models is the attempt to use traditional transactions as building blocks which restricts the applicability in the workflow domain. Workflow activities are in general not database transactions which are started automatically and therefore the concepts of advanced transaction models cannot be applied directly. Major work in expanding advanced transaction models for workflow requirements was done in the area of transactional workflows (e.g. [RS93, Hsu93, BDS+93]) and long-running activities (e.g. [DHL91, WR92]). Nevertheless, this work still is mainly influenced by a database point of view and therefore not direct applicable for workflow systems.

Modern WFMSs have to support complex, long-running business processes in a heterogeneous and/or distributed environment. It has been pointed out in [GHS95] that most of these systems lack the ability to ensure correctness and reliability of workflow execution in the presence of failures. Therefore a strong motivation of merging advanced transaction models with workflow models becomes evident. "*Workflow Transactions*" [EL95, EL96b] incorporate the basic ideas of traditional transactions extended by several features of advanced transaction models required for workflow execution. Currently there are several approaches reflecting this idea (e.g., [KS95, MAA+95, AKA+95b, Ley95]). The main characteristics of workflow transactions may be summarized as follows:

- *Analogies:* A flat transaction is a sequence of operations which transfers a database from one consistent state into another consistent state. In analogy to this, a workflow transaction is a *sequence of workflow activities* which transfers a business process from one consistent state into the next consistent state. From this point of view, we talk about workflow transaction on a more abstract level than we are used to talk about traditional transactions.

- *Transaction structure:* Workflow transactions must allow a hierarchical structure in order to be applicable for complex applications. A workflow typically consists of several activities which themselves again may be composed of (sub-) activities. Each activity is a workflow transactions. The nesting must be supported over as many levels as there are abstraction layers in the application. The most elementary activities finally represent an application program, a flat transaction or for example a human task (e.g. making a phone call).

- *Atomicity:* Since workflow activities are in general of long duration, application dependent (user-defined) failure atomicity is required. The goal is not to undo everything in case of a failure but instead to *selectively roll back* parts of the work until the most recent *consistent* state (within the transaction) is reached. In order to find such a consistent state the workflow transaction manager needs support from a human expert (e.g. in WAMO [EL95] such states can be identified during runtime, based on specific information given by the workflow designer). Additionally, it would not be a realistic option in online systems to go back to the past (which corresponds to a rollback operation). Instead, more advanced recovery concepts [EL96b, Ley95] (especially compensation of already committed activities) have to be provided. Having reached such a point, forward execution of the process - perhaps on an alternative path - must be supported. Summing up, workflow transactions relax the "nothing" property of the all-or-nothing concept of traditional transactions.

- *Consistency:* As for traditional transactions, the scope of consistent executions does not focus on the work which is done within an activity (a transaction) but only on the correct execution order of activities. The commit of an activity is taken as a guarantee that the activity has produced a consistent result. If an activity aborts (fails) then an inconsistent state may be the consequence. As for flat transactions, such situations should not occur and must therefore be removed - ideally automatically. If compensating activities are involved in a recovery process then this activities have to terminate successfully in order to preserve consistency.

- *Isolation:* Because of the nature for workflow applications (long duration, cooperation, concurrency) it is not possible to execute workflow transactions fully isolated from concurrent transactions. As mentioned earlier, serializability as correctness criterion for concurrent processing is too restrictive. There exist several theoretical approaches to overcome this problem without compromising consistency, as for example semantic serializability [BDS+93]. The goal is to exploit the semantics of the activities (by a human expert) by defining compatibility specifications between the activities. Compatibility between two activities means that the ordering of the two activities in the schedule is insignificant from an application point of view.

  Additionally, isolation is relaxed in the sense that sub-activities externalize their results as soon as they commit in order to increase concurrency and hence performance. Of course, if the parent activity later on aborts or is involved in a compensation process then the results of the activity may have become invalid and therefore the activity has to be undone semantically (compensated). At this point it must also be mentioned that a compensation of activities is not always possible (e.g. drilling a hole cannot be undone later on). Irreversible activities often are called “critical” activities [EL95].

- *Durability:* As soon as a workflow (sub-)transaction commits, its effects are persistent. Of course, these effects may be semantically undone later if the parent activity aborts or is compensated.

Summing up, workflow transactions are quite different to traditional transactions. One important aspect is that each workflow applications has its own correctness and reliability requirements which have to be satisfied by a corresponding *extended or customized transaction model* (ETMs). Consequently, such a transaction model has to be very flexible and adaptable to different workflow application-specific requirements. Additionally, we believe that transaction specific features already have to be available at workflow design level in order to express certain business process consistency requirements and to exploit the possibilities of workflow transactions.

6 Conclusions

Workflow systems are developed to meet the needs of modern business organizations. A variety of different functionalities are offered supporting different types of processes from human centered to machine centered processes. Document-oriented and process-oriented workflows could be identified as the major types of workflow systems.

Workflow management system are also interesting from a technical point of view. They can serve as platform for the integration of various different applications in particular they can provide the necessary functionality to combine isolated legacy systems and integrate them in a modern information infrastructure.

Databases are important building blocks for workflow management systems. They manage case relevant as well as process relevant data in a reliable, efficient and consistent way. By mapping the workflow description to triggers, an active database management system can serve as workflow engine. The transaction concept of databases, however, is too rigid to apply directly for workflow systems. Nevertheless, it offers the basic functionality for developing and implementing workflow transaction concepts which take care of consistent execution of business processes and provide intelligent, automatic treatment of errors and exceptions.

BIBLIOGRAPHY

[AKA+95a] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör, and C. Mohan. Advanced Transaction Models in Workflow Contexts. Technical report, IBM Almaden Research Center, 1995.

[AKA+95b] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör, and C. Mohan. Failure Handling in Large Scale Workflow Management Systems. Technical report, IBM Almaden Research Center, 1995.

[BDS+93] Y. Breitbart, A. Deacon, H.-J. Schek, A. Shet, and G. Weikum. Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows. *SIGMOD RECORD*, 22(3):23–30, Sep. 1993.

[BJ94] Christoph Bußler and Stefan Jablonski. Implementing Agent Coordination for Workflow Management Systems Using Active Database Systems. In *[WC94]*, pages 53–59, 1994.

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

[CKO92] B. Curtis, M.I. Kellner, and J. Over. Process Management. *Communications of the ACM*, 35(9):75–90, 1992.

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

[DHL91] U. Dayal, H. Hsu, and R. Ladin. A Transactional Model for Long-Running Activities. In *Proc. of the 17th Int. Conference on VLDBs*, Barcelona, September 1991.

[EB82] C. A. Ellis and M. Bernal. Office-Talk-D: An Experimental Office Information System. In *Proc. First ACM-SIGOIS Conference on Office Information Systems*, pages 131–140, 1982.

[EGN94] J. Eder, H. Groiss, and H. Nekvasil. A Workflow System Based on Active Databases. In G. Chroust and A. Benczur, editors, *CON 94: Workflow Management: Challenges, Paradigms and Products*, pages 249–265. Oldenbourg, Linz, Austria, 1994.

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

[EL95] J. Eder and W. Liebhart. The Workflow Activity Model WAMO. In *Proc. of the 3rd Int. Conference on Cooperative Information Systems*, Vienna, Austria, May 1995.

[EL96a] J. Eder and W. Liebhart. A workflow classification framework. Technical report, University of Klagenfurt, Department of Informatics, January 1996.

[EL96b] J. Eder and W. Liebhart. Workflow Recovery. submitted for publication, January 1996.

[Elm92] A.K. Elmagarmid. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann Publishers, 1992.

[GE93] Herbert Groiss and Johann Eder. Einsatz von Aktiven Datenbanken für CIM. In *Tagungsband des ADV-Kongresses Informations- und Kommunikationstechnologie für das neue Europa*, pages 861–870, 1993.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Shet. An Overview of Workflow Management: From Process Modeling to Workflow Automation. In A. Elmagarmid, editor, *Distributed and Parallel Databases*, volume 3, pages 119–153. Kluwer Academic Publishers, Boston, 1995.

[GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[Gro92] Object Management Group. The Common Object Request Broker: Architecture and Specification, 1992.

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

[Hsu93] M. Hsu. Special Issue on Workflow and Extended Transaction Systems. *Bulletin of the Technical Committee on Data Engineering*, 16(2), June 1993.

[IA95] ISO and ANSI. Working Draft Database Language SQL (SQL3), March 1995. X3-H2-95-084.

[Jab94] S. Jablonski. Functional and Behavioral Aspects of Process Modelling in Workflow Systems. In G. Chroust and A. Benczur, editors, *CON 94 Workflow Management: Challenges, Paradigms and Products*, pages 113–133. R. Oldenburg, 1994.

[KS95] N. Krishnakumar and A. Shet. Managing Heterogeneous Multi-System Tasks to Support Enterprise-Wide Operations. In A. Elmagarmid, editor, *Distributed and Parallel Databases*, volume 3, 1995.

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

[Ley95] F. Leymann. Supporting business transactions via partial backward recovery in workflow management systems. In G. Lausen, editor, *GI-Fachtagung: Datenbanksysteme in Büro, Technik und Wissenschaft*, Dresden, Germany, March 1995. Springer Verlag.

[MAA$^+$95] C. Mohan, D. Agrawal, G. Alonso, A. El Abbadi, R. Gnthr, and M. Kamath. Exotica: A Project on Advanced Transaction Management and Workflow Systems. *ACM SIGOIS Bulletin*, 16(1), 1995.

[McC92] S. McCready. There is more than one kind of Workflow Software. *Computerworld*, 2, November 1992.

[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.

[RS93] M. Rusinkiewicz and A. Shet. On Transactional Workflows. *Bulletin of the Technical Committee on Data Engineering*, 16(2), 1993.

[RS94] M. Rusinkiewicz and A. Shet. Specification and execution of transactional workflows. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*. ACM Press, June 1994.

[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.

[WC94] Jennifer Widom and Charma Chakravarthy, editors. *4th Int. Workshop on Research Issues in Data Engineering*, February 1994.

[Wor94] Workflow Management Coalition, Brussels, Belgium. *Glossary: A Workflow Management Coalition Specification*, November 1994.

[WR92] H. Wächter and A. Reuter. The Contract Model. In A.K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, 1992.

[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.