The WASA Approach to Workflow Management for Scientific Applications

Gottfried Vossen and Mathias Weske

Lehrstuhl für Informatik, University of Muenster
Grevener Strasse 91, D-48159 Muenster, Germany
{vossen,weske}@helios.uni-muenster.de

Summary. Workflow management has gained increasing attention recently, since it allows to combine a data-oriented view on applications, which is the traditional one for an information system, with a process-oriented one in which activities and their occurrences over time are modeled and supported properly. While workflow management has mostly been considered in business applications so far, the focus of the WASA project is on *scientific* applications such as geoprocessing, molecular biology, or laboratory environments. In particular, WASA aims at flexible and platform-independent workflow support, both w.r.t. specification and execution of workflows. It turns out that the modeling and execution of workflows in traditional and in scientific applications exhibit significant differences. In particular, the need for dynamic modifications of workflow models while workflows are running is an important feature in scientific applications. Observations like these have resulted in a generic WASA architecture, which can be tailored towards various specific application domains. The conceptual design and functionality of a WASA prototype is outlined, in particular that of its core workflow engine, and it is shown how the requirements of flexibility in modeling and executing workflows, imposed by scientific applications, are met by this prototype.

1. Introduction

Workflow management aims at modeling and controlling the execution of processes in both business applications [11, 18] and scientific applications [14, 26, 33, 25]. It has gained increasing attention recently, since it allows to combine a data-oriented view on applications, which is the traditional one for an information system, with a process-oriented one in which (human or machine) activities and their occurrences over time are modeled and supported properly [30]. While a number of workflow management systems for business applications are commercially available already, systems for scientific applications are still in their infancy. A major goal of the WASA[1] project is to remedy this situation. Specifically, WASA tries to take the particular requirements of these applications, such as high modeling and specification flexibility as well as platform independence [2, 20], into account. In this paper we survey the WASA project. In particular, we characterize the specific properties of scientific workflows as opposed to business workflows. We develop a generic WASA architecture and describe the conceptual design and functionality of a prototypical implementation.

---

[1] Workflow-based Architecture to support Scientific Applications

In business applications, workflow management is of increasing strategic importance. Indeed, the identification of *business objects* and the process of *re-engineering* business procedures is considered highly relevant to the future development of commercial enterprises. On the other hand, the transition from data modeling to process modeling appears relevant in a variety of non-business applications as well. Indeed, there is an increasing number of experimental sciences that rely on computers and software, and that needs guidance through the appropriate exploitation of this technology; as it turns out, workflow management can provide such guidance. The applications we refer to include experimental physics, molecular biology, and geoprocessing; more generally, we are interested in applications in which experiments are done in a laboratory environment and hence the experiments themselves as well as the information they consume or produce need to be managed properly. Surprisingly, there are not yet too many system developments which try to take care of these domains. The systems we have identified generally fall into the following categories:

1. Environments that are tailored towards a specific application domain: OASIS [21], ZOO [15, 1]. Application specific environments built on top of database systems are Moby Dick [26] or XBio [17].
2. Workflow management systems primarily for traditional (business-type) applications which could be used in scientific environments, although the latter were not foreseen as targets initially: Mentor [35], Meteor$_2$ [27], Mobile [16, 4], and Panta Rhei [8]; clearly, commercial systems such as IBM's FlowMark [18, 13] also fall into this category.
3. "Workflow-aware" systems for scientific applications: LabBase [28], CRISTAL [19], MapBase [12].

OASIS is an environment for data analysis, knowledge discovery, visualization, and collaboration, and is directed towards geo-applications. Its implementation is based on a CORBA-compliant distributed object management system. The geo-sciences are also a target of the WASA project; however, the WASA project focuses on support for pre-defined processes (which may change over time), while the OASIS project lays an emphasis on a loose cooperation of agents to achieve a common goal. ZOO is essentially a software package that allows scientists to manage experiments and data related to an experiment from a desktop machine through a uniform interface. However, ZOO emphasizes data modeling and experiment modeling at a high language level and at an individual basis, and ignores workflow aspects; we try to prove that a process orientation right from the beginning is a more appropriate way to go.

Besides OASIS, several other system developments also exploit CORBA technology and hence distributed object management facilities. This particularly applies to experimental workflow management systems such as Meteor$_2$ and Mentor appearing in the second category. In Meteor$_2$, workflow programs are generated from workflow specifications, and CORBA is used for execut-

ing workflows in distributed environments. Mentor incorporates a CORBA-compliant broker to integrate external applications into a workflow execution environment. Here the major development goals include good scalability, high availability, heterogeneity, and distributed workflow executions which even allow for formal verification [35]. In our project these aspects, although important, are currently not an issue, since our emphasis is different. The idea of using a Web browser as end-user interface pops up in the next two systems mentioned in Category 2, Mobile and Panta Rhei. Mobile takes the Web as a service for building user interfaces and for integrating or implementing external applications, while Panta Rhei primarily uses the Web for exchanging forms between an end-user, i.e., a person involved in a workflow execution, and the workflow engine. The difference to WASA is that our system is implemented entirely in Java and hence not only usable through any ordinary Web browser, but also vastly machine-independent and open to all kinds of enhancements brought along by the Java world. Finally, commercial workflow tools such as FlowMark are, as has been argued in [3], an obstacle to dynamic workflow modifications, as they are typically based on a built-time/run-time-approach; as a consequence, specifications must be complete before they can be executed, and changes always require a halt followed by a system restart.

The developments that we consider closest to our project are those in the third category. Indeed, LabBase and CRISTAL also try to bring workflow concepts into scientific applications. LabBase is a system for managing workflows in large semi-automated laboratory projects which sits on top of the MapBase database system. Its workflow manager essentially controls the execution of laboratory protocols in which experiments to be conducted are described. Protocols may change frequently, since the sequencing or composition of their experiments are altered. The workflow manager is able to take care of this by taking appropriate input from the respective application at its interface; basically, it is programmed from outside for each experiment that is executed. As will be demonstrated, WASA takes a significantly different route; it does not have an API for workflow control, but a dedicated workflow engine that responds to database updates representing dynamic changes in workflow models. Finally, the CRISTAL system (Concurrent Repository and Information System for the Tracking of Assembly Lifecycles) emerged from the Compact Muon Solenoid (CMS) experiment in particle physics [19]. The preparation and instrumentation of that experiment evolves dynamically, mostly as the result of preliminary tests that are made, and needs to integrate resources from remotely located research centers. Here the approach is to employ product data management tools, which are able to organize and control product data as well as product life-cycles, and to extend these by workflow management capabilities for being able to keep track of production system activities. Again, this is different from the WASA approach, since workflow management comes in as an aid to product data management, while

we consider databases or files subordinate to workflow engines and emphasize activity descriptions and control.

The paper is organized as follows: In Section 2. we discuss the commonalities and differences of workflow modeling in traditional applications and in scientific applications, and we present a case study of a scientific workflow from the area of molecular biology. Section 3. describes a key property of scientific workflows, namely dynamic change of workflow models while workflows run. The WASA architecture is presented in Section 4.; in Section 5. we discuss the conceptual design and the functionality of a prototypical implementation of the WASA workflow engine. Concluding remarks in Section 6. complete the paper.

2. Workflow Modeling

In this section we discuss specific properties of the workflow modeling process and of the resulting workflow models in business applications and scientific applications. In particular, we discuss properties of scientific workflows, and provide a set of features which are present in scientific workflows but which are not present in business workflows. To support our statements, examples from the areas of molecular biology and laboratory information systems are provided. These observations will serve as a motivation for two basic dynamic change operations that are useful for the application areas in question. These operations are discussed in Section 3.

2.1 Workflow Modeling in Business Applications

So far, the major application area of workflow management has been the business field [11, 18, 30]. In particular, modeling and re-engineering of business processes has become a strategic goal in many enterprises, with the final goal of enhancing flexibility and efficiency of business processes and thereby improving customer satisfaction as well as enterprise performance.

Workflow applications are developed in complex processes, roughly characterized as follows. After initial information gathering on the relevant business procedures, the modeling and specification of business processes is performed. When this phase is completed, the resulting process model is transferred into a workflow model, which is a computerized representation of activities and their execution relationships used by a workflow management system for the controlled execution of workflows [11, 29]. Various tools are now commercially available which support the development of workflow applications. Business process modeling and re-engineering tools focus on the early phases, while workflow management systems support the later phases of the workflow development process.

Specific properties of application processes have implications for the required functionality of a workflow management system. In general, business processes to be modeled and analyzed are usually well understood. The major aim of modeling is to find bottlenecks with the goal of increasing efficiency and reducing cost of doing business. This means once the business modeling phase is completed, a workflow model is created to implement the process. It is important to notice that the structure of the workflow model is fixed and does not change over time. Workflow models with this property are called *static*. Static workflow models might be changed during future re-engineering processes, which then also involve earlier phases, namely business process modeling. In this context, changes to the structure of workflows are not permitted during workflow executions; they usually require a re-compilation with subsequent re-execution.

Since static workflow models are well-suited to describe most business processes, current workflow management systems provide appropriate support for controlling workflows described by static workflow models. These are brought onto the workflow management system, which is used to control the executions of many (often hundreds of) workflow instances of a given static workflow model.

We next focus on scientific applications and show that their requirements render workflow management systems unable to realize dynamic changes inappropriate for the controlled execution of scientific workflows.

2.2 Workflow Modeling in Scientific Applications

As mentioned before, new applications for workflow management are emerging, among which scientific ones seem to play a major role [33, 3, 19]. Workflows in these domains differ significantly from business workflows. This is mainly due to the fact that concepts in scientific applications are not as stable, and processes often are not completely known in advance. The lack of complete knowledge about the processes in scientific applications has implications for the modeling of scientific workflows. The main issue is that workflow models are inherently incomplete, or they may even change over time, either in predictable or unpredictable ways. We discuss several examples where these properties show up before presenting a case study of a scientific workflow from the domain of molecular biology.

– *Order Processing in Laboratory Environments:* Consider a chemical analysis enterprise, which processes orders for sample analysis. Assume an order represents the chemical analyses of a number of samples, ranging from soil to crops and animal feed. The high volume (10^6 samples per year) makes workflow management useful to enhance throughput. The samples are processed independently within an order workflow. For each sample there is a default structure of the steps to be carried out, i.e., of the workflow to be executed for processing the sample. This default structure can change. A

common form of change occurs when the customer wants additional analyses performed on a given sample, or wants some analyses redone to validate the results.

These changes cannot be foreseen completely, they are vastly *ad-hoc*. Hence, changing the workflow dynamically is needed to (i) be able to use the controlled execution of workflows to enhance efficiency of throughput while (ii) supporting dynamically changing procedures. The situation characterized can be coped with by a feature called *dynamic modification*, which means the ability of a workflow management system to change workflow models while workflows of this model are running.

– *Experiment Design:* A common property of scientific experiments is incomplete specification. In particular, the activities of some initial part of an experiment are typically known before the experiment starts. However, the researcher decides on the continuation of the scientific experiment only after execution (or even a preliminary evaluation of the results) of the first part of the experiment, i.e., after the completion of the partial workflow. In this case, the point in time when the decision is taken may be available before the workflow starts. Thus, a dynamic change of the workflow model is now known *a priori*.

2.3 Case Study: Scientific Workflows in Molecular Biology

We now provide a specific example of an application in which WASA is relevant and its use appropriate – the area of *DNA Sequencing*. We give a brief introduction to the field and describe how to perceive DNA sequencing as a workflow problem. For further details on the material presented in this section, the reader is referred to [20]; a complete report on our findings regarding the use of a commercial product is [3].

The issues dealt with in DNA Sequencing can roughly be summarized as follows. Recall that all genetic information of organisms is stored in nucleotide sequences. One form is desoxyribonucleid acid (DNA), which consists of two parallel strands, each of which is a sequence of bases, identified by A, C, T, and G. Finding and interpreting the base sequence of an organism is an important and fundamental task in molecular biology [10]. Today, short sequences of DNA (≤ 500 base pairs) can be generated semi-automatically, using specific devices; these sequences are known as *fragments*. Among the complicating factors of DNA Sequencing are *errors* (device inaccuracies and spontaneous mutations), *contamination*, *lack of coverage*, or *repeats* (long, repeated strings in the target DNA sequence). Because of these inaccuracies, the base sequence returned by a sequencing device may not be entirely correct; it is therefore considered as *raw data*. One task of scientists is to correct the errors in such a sequence, and to produce data sets of higher quality. This is done using predefined validation procedures as well as the scientist's expert knowledge.

Since direct sequencing is possible only for relatively small DNA fragments, a divide-and-conquer strategy has to be used in order to determine the base sequences of larger stretches of DNA (e.g., 10Kbase pairs). It is such strategies that are known as *fragment assembly*; they basically consist of the following steps:

1. Generate multiple copies of DNA (cloning),
2. cut clones into pieces (fragments),
3. sequence fragments directly,
4. assemble fragments.

After sequencing and validating a number of fragments, the assembly process starts (note that, due to resource limitations, typically not all fragments can be sequenced). Sequences that consist of a number of assembled fragments are known as *contigs*. Having produced a number of contigs, and trying to find a place for the next fragment, there may be zero, one, or more places to put it; the latter may happen due to repeated sequences in the genome. To solve ambiguities of this kind, or to fill remaining gaps, new fragments have to be sequenced. The information on the next steps to take becomes available only while the experiment is being conducted.

We now look at the DNA sequencing process and its subprocess "fragment assembly" and specify them using workflow terminology. The underlying idea of our approach is the following. Fragment assembly tools are typically programs composed of many modules that have to perform the various steps mentioned in the previous subsection. However, since there is not a unique accepted strategy for doing fragment assembly, people perform several experiments, and do so, among other things, by configuring distinct program modules into new assembly programs. It is here that workflows come into play: we consider that each such configuration activity corresponds to specifying a workflow. In other words, composing a fragment assembly tool from given modules is treated as a workflow specification. Consequently, the workflow management system used executes the resulting program system, hence driving the various modules it comprises. Moreover, it can feed input and receive output data to and from the various assembly steps, and may even be capable of triggering external devices.

To specify workflow models, we use a simple graphical notation based on directed graphs, whose nodes represent tasks, and whose edges represent relationships between tasks, e.g., control flow. Tasks can be nested in an arbitrary fashion, by unfolding nodes into subgraphs. The overall structure of the DNA Sequencing workflow, i.e., the top-level workflow, is shown in Figure 2.1; each task shown will undergo one level of refinement.

The *initial generation* task specifies the scientific experiments that aim at extracting sequence information from a given chemical molecule. This usually consists of a number of steps that result in the production of a film from which the base sequence of the molecules under consideration are read and
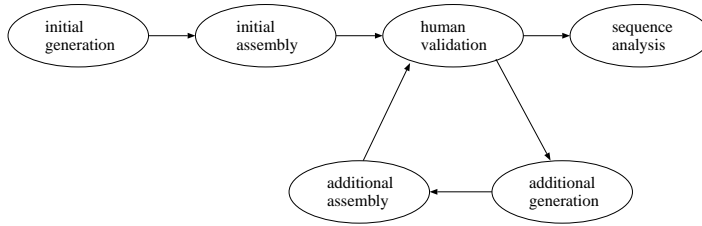
Fig. 2.1. DNA Sequencing seen as a Workflow.
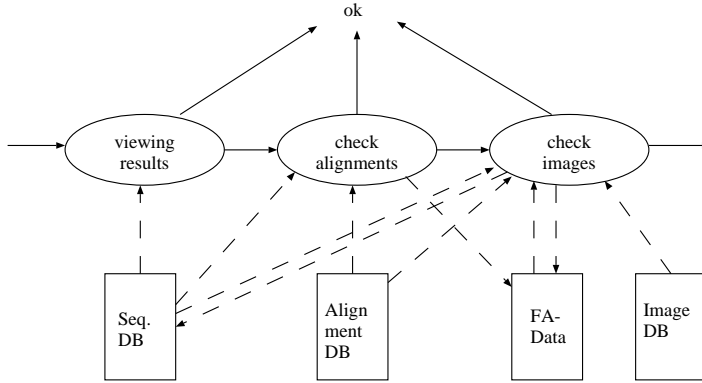
Fig. 2.2. The Step of Human Validation.

entered into a sequence database. *Initial assembly* uses the set of fragments generated by the first step and tries to assemble them to generate the desired sequence. The result of this activity is validated by a human. If satisfied, *sequence analysis* can be launched, which runs the sequence found against one or more community databases, and performs various analysis tasks, like examining the sequence for the presence of genes.

Each of these global activities is refined in [20]. We here restrict our discussion to the *human validation* activity, as shown in Figure 2.2. This activity is especially important whenever two or more consensus sequences are generated during initial assembly, since the goal of the assembly is to return a single sequence, not a number of partial sequences of the sequence under consideration. Human validation starts with viewing the results of the assembly process.

If the assembly produced a single consensus sequence that has a high percentage of correct overlaps, then the human can decide whether the assembly process was successful and thus that the next task of the top-level workflow can start. In Figure 2.2 this is indicated by an edge to "OK". If, however, the human is not satisfied, certain alignments (matching of fragment patterns) have to be checked. In order to do so, the human accesses sequences stored

in the sequence database as well as the set of alignments that were used to perform the alignment. Looking at the alignments may render the objections obsolete, again leading to "OK". However, the scientist may suspect an error in the task that led to the incorrect alignment. In this case, the *check images* task is performed. In this task, the human accesses the film images stored in the image database and fragment assembly data (FA-Data) to validate the reading process or to detect reading errors. To decide for the latter, accessing the sequence database and the alignment database is necessary, in which case the loop of the top-level workflow is iterated and additional fragments are generated and added to the consensus sequences found so far. The loop is exited when a human validates the results, in which case the consensus sequence found is analyzed. As already mentioned, further refinements of the workflow model shown in Figure 2.1 can be found in [20].

3. Dynamic Change Operations

We next identify two forms of dynamic change operations to cope with the issues stated above. In Section 5. we will explain how our prototype is designed to support the dynamic change operations required, and how these can be executed by the user. (A more detailed discussion of flexibility issues in workflow systems appears in [32].)

3.1 Anticipated Dynamic Change

The first form of dynamic change deals with *anticipated, predicted* dynamic changes. Operations of this form appear whenever the modeler of a workflow knows the exact position at which a modification of the workflow model might appear (and will be appropriate). In this case the workflow model explicitly includes modeling activities which are executed within the workflow. Anticipated dynamic changes are necessary when there is no complete knowledge of the workflow before the workflow starts, i.e., if the workflow model is incomplete. To cope with this situation, the modeler explicitly specifies a modeling activity as one activity of the (incomplete) workflow model (typically the one that completes the specified part of the workflow). During the execution of the workflow, the modeling activity is started. When the modeling activity starts, the system offers a number of sub-workflows to implement the activity, which were defined beforehand. The modeler may choose one of the pre-defined sub-workflows to implement the activity by accessing the workflow library. However, if none of the pre-defined workflow models seems suitable to continue the workflow properly, he or she may decide to specify a new workflow to implement the activity. In this case, workflow modeling is done *while the workflow executes*. Finally, the chosen workflow is started, and the workflow execution resumes.
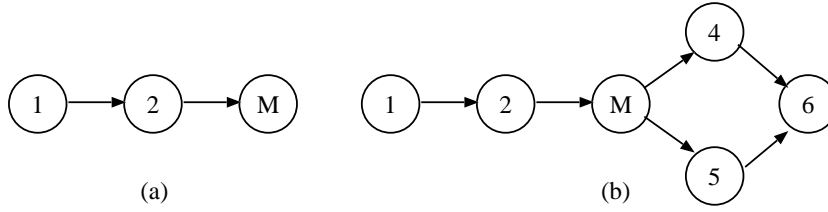
Fig. 3.1. Anticipated Dynamic Change, Implemented by Modeling Activity M.

Anticipated change of workflow models is a common property of scientific workflows [33], where scientists decide on future experiments only after initial experiments have been performed and − at least partially − evaluated. The modeling activity of the scientific workflow corresponds to choosing an experiment to execute next, or to designing a new experiment. A simple and abstract workflow model including a modeling activity is depicted in Figure 3.1(a). The incomplete workflow consists of activities 1, 2 and a modeling activity M. During the modeling activity the agent performing it decides to continue the incomplete workflow with the concurrent execution of activities 4 and 5, followed by an activity 6; the workflow model that results from the dynamic modification is given in Figure 3.1(b). The final activity of an incomplete scientific experiment is a modeling activity which involves choosing activities to continue the scientific workflow.

Note that in a situation like the one just described, workflow parts will typically be known in advance, as will be rules that guide the process of combining them into larger workflows.

3.2 Ad-hoc Dynamic Change

The second form of dynamic change deals with *ad-hoc*, i.e., not anticipated dynamic changes. It is characterized by the need to change the future behavior of a running workflow at an unpredicted point in time during the execution of the respective workflow.

This situation has commonalities with exception handling in traditional programming. However, in programming the programmer defines a set of situations for which an exception is raised, and a set of procedures to be executed when the exception is raised. In contrast, ad-hoc changes of workflow models do not need to be specified *a priori*. In many applications, specifying all possible reasons for an ad-hoc change is a cumbersome if not impossible task. Moreover, modelers do not want to model situations that may or may not happen in the future. Hence, support for these kind of exceptions by ad-hoc changes is important for a flexible workflow system.

In ad-hoc change, a modeling activity can be started at any time during the execution of a workflow. An example from the area of scientific workflows
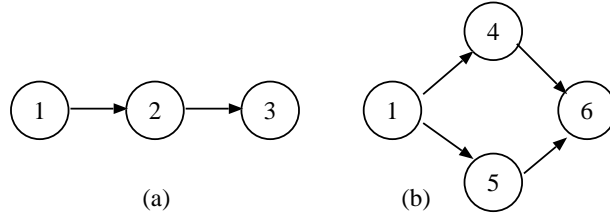
Fig. 3.2. Ad-hoc Modification.

is a complex experiment that involves numerous persons and resources, for instance specialized devices. When, during the execution of a scientific experiment, some devices or chemical agents are not available and the experiment cannot be delayed until the devices are operational again, there has to be an ad-hoc change of the model of the scientific workflow. For instance, additional clean-up activities have to be started to shut down the experiment properly. We mention that ad-hoc changes are not limited to scientific experiments. In certain business applications, ad-hoc changes are also required [9], with the aim of saving modeling effort and providing high flexibility for the application by allowing for unpredicted, ad-hoc changes.

A sample workflow model with an ad-hoc change is given in Figure 3.2. Assume a workflow has been started based on the workflow model given in Figure 3.2(a). After executing activity 1 an ad-hoc change occurs. The person responsible for the workflow decides not to continue the workflow as specified (continue the workflow with activities 2 and 3) but to execute activities 4 and 5 concurrently before completing the workflow with activity 6. The resulting workflow model is given in Figure 3.2(b). Notice that the workflow model does not contain any explicit modeling activities as in the previous example.

The situation just described occurs, for example, in order processing applications in laboratory environments, as discussed above. The ad-hoc change is usually initiated by a customer who decides to add some analysis procedures to an order. This change may occur at any point in time while the workflow runs, i.e., while the samples of the order are analyzed. Returning to the example of Figure 3.2, instead of performing, analysis 2 and 3 on a given sample, the customer decides that he wants analyses 4 and 5 performed concurrently before analysis 6 is done. (Performing analyses concurrently is attractive since analyses may take days or even weeks.) In this scenario, the dynamic change should only apply to the workflow instance under consideration; other workflows should not be affected by that dynamic change.

3.3 Related Work on Dynamic Change

In terms of flexibility in modeling and executing workflows, our approach relates to several approaches that have been reported in the literature.

This does not come unexpected, as workflow management combines influences from a variety of disciplines, including cooperative information systems, computer-supported cooperative work, groupware systems, and active databases [30]. Ellis et al. [9] show that the need for dynamic change is present in office workflows, in order to circumvent or augment standard procedures. As stressed by Craven and Mahling [7], work environments nowadays are becoming increasingly dynamic, which requires new types of support for cooperative work. This motivates that, while these properties are important for workflow management in scientific applications [14, 33], modern requirements in business computing also benefit from them. In particular, specifying error conditions may not be feasible in business applications, and dynamic change of workflow models is an important functionality of a workflow management system to cope with exceptions [9]. Reichert and Dadam present $ADEPT_{flex}$, an approach for controlled dynamic modifications of workflow specifications based on non-nested, symmetric workflow specifications [24].

We conclude this section by mentioning that dynamic modifications raise a number of questions, ranging from modeling issues to system requirements for an implementation of this concept. One question addresses the workflow instances which should be effected by a dynamic change. Changes of workflow models may apply to a single workflow instance, to multiple workflow instances, or to all workflow instances of a given workflow model. An example of the latter is the adjustment of a workflow model to the change in the environment of the process, e.g., the installation of a new laboratory procedure which is obligatory for all scientists of a given laboratory. In this case it is very important that all future workflow instances reflect the changes. Furthermore, the active workflow instance of this model which are not yet executing the changed part of the workflow should also be affected.

4. The WASA Architecture

In this section we describe the WASA architecture, whose overall goal is to provide an integrated, workflow-based environment for scientific work.

4.1 The WASA Layers

In the WASA architecture, the following four layers can be distinguished (see Figure 4.1):

1. User Interface Layer
2. Internal Tools Layer
3. Enhanced Database Functionality Layer
4. Database Layer

The users for which this environment has been designed are expected to be scientists which are experts on some domain, and to have experience in designing new experiments and controlling the execution of existing ones. Also, users will be supposed to have a basic familiarity with the workflow paradigm, so that they are able to express their needs and plans in a corresponding formalism. A general assumption underlying the WASA architecture is that all data manipulations are done via the WASA interface.

The *User Interface Layer* is responsible for communication between users and the rest of the system. It consists of four main functional blocks which communicate with each other (and with the internal tools, see below). The *Specification and Design* facility provides users with tools to specify and design experiments by means of workflows; in addition, it is intended to provide access to previously designed workflows for re-use, to support shared workflow design, and to allow the configuration of partially specified workflows into a new one. The *Data Manipulation* facility provides users with means for accessing and updating data concerning applications and experiments. This includes navigation through reported experiments and their results as well as the invocation of analysis procedures (e.g., through method execution calls). This module would typically encapsulate facilities such as those provided by a database query language processor.

The *Browsing and Visualization* module allows users to browse and visualize different kinds of application data as well as management data. It allows access to application specific data as well as to general data. Clearly, visualization is highly domain-specific. For example, in molecular biology, visualization of DNA sequences and of protein structures may be relevant; in medical experiments, physicians may need to visualize 3D renderings of objects; in geography, maps of different resolution and levels may need to be displayed. Finally, the *Runtime Monitor* allows users to execute previously defined workflows, and to monitor and control their executions.

The *Internal Tools Layer* consists of the workflow management system and of a set of auxiliary managers to support experiment specification, documentation, and execution. The *Workflow Engine* is the core component of the WASA system. It provides the core functionality for scientific workflow management. The *Documentation* manager provides the means to document the conduction of an experiment as well as its specification. It also allows recording of relevant events that occurred during the specification or execution of the experiment. The *Analysis* manager is responsible for managing the interface to application-specific analysis procedures, and for controlling their execution as requested by users. We imagine this to be a loose coupling only, in the sense that this manager will generally know where to find relevant procedures and analysis tools, to export input data to such tools, and to import their results and findings into the WASA system. The *Decision and Planning Support* manager helps to guide users in designing and conducting experiments. Decision support on how to continue a given exper-
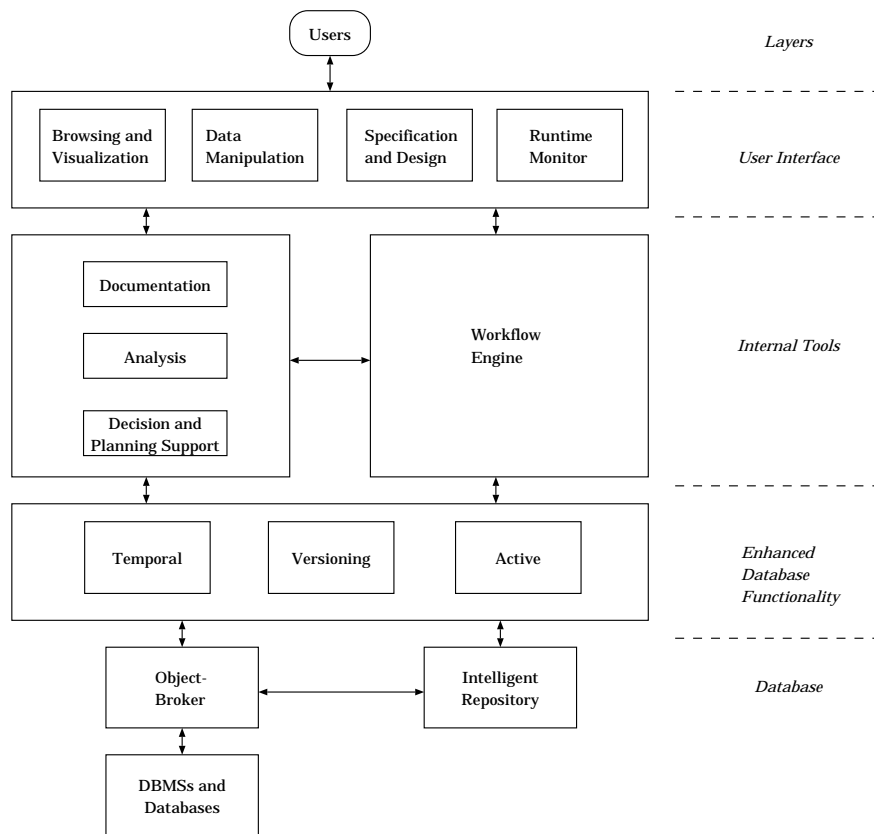
Fig. 4.1. The Architecture of WASA.

iment would also be a task of this component, i.e., which step to take next in a given experiment. Thus, it contains knowledge that is domain specific, as well as knowledge about workflow construction. This is extremely relevant to situations in which an experiment starts by executing a partially specified workflow, but where its continuation and successful completion will depend on intermediate results and decisions based on these results.

Scientific experiments generally need the capability to manage a variety of data using different types of facilities. Although the WASA architecture is based on multiple database systems which can be incorporated as data sources, we cannot assume that all these database systems will provide advanced functionality; indeed, it is common nowadays to build scientific applications on legacy systems such as relational databases, or data files. As a consequence, the enhanced database management layer of WASA consists of a set of mechanisms that provide the relevant advanced functionality in a system-independent way. We consider three types of functionality of particular importance here: (i) Being able to handle *temporal* aspects, (ii) being capable of distinguishing versions (of both data and workflow models), and (iii) being able to support *active* features for controlling the execution of a given task or workflow. In order to make this layer portable and independent of pre-existing databases in particular application domains, WASA is intended to provide access to individual databases through a standardized interface. To this end, it is reasonable to assume that underneath WASA there is typically a federation of multiple databases with a common interface such as an object request broker [22, 23] on top. This broker is responsible for retrieving the appropriate data and control descriptions from stored data sets, and is expected to support a variety of data models and query languages.

As described above, the underlying databases which store and provide relevant data are not part of the WASA architecture; instead, they will be handled through the object broker interface. However, the system creates numerous internal data and information, which is relevant, for example, to the various workflows it handles. So it is clear that the database layer of WASA has to contain two categories of data and, hence, two types of databases: (i) data used by applications, which is supposed to be stored in a number of databases whose details are hidden from the system through the broker, and (ii) an intelligent data repository, i.e., a WASA-specific database, which is used to store data needed to run the upper levels of the architecture (e.g., constraints on workflow construction, documentation) as well as special structures which are used by the database layer to perform its functions.

5. The WASA Prototype

In this section we discuss the conceptual design of a WASA prototype, its system architecture and functionality. We first focus on modeling capabilities and on dynamic modification operations, as discussed earlier. The system

functionality also includes support for further aspects, such as the organizational aspect used for role resolution. The prototype allows to execute workflows based on workflow models, including management of work-item lists of workflow clients and automatic start of applications to perform activities within workflows. We discuss the architecture of the system and the functionality of its components below.

5.1 General Considerations

The traditional approach to workflow management is based on the "built-time/run-time"-paradigm: During the built-time workflow models are specified completely. Complete workflow models are used at run-time to control the execution of workflows. Since dynamic modification requires more flexibility than this approach has to offer, we believe that an interpretational approach is more feasible. In this approach, the system interprets workflow models successively, whereas in the compilation approach the whole specification of the workflow is transferred into an executable format, and only then workflow executions are started. The step-by-step interpretation allows more conveniently to react to unpredicted changes in the application. The prototype is based on the interpretation approach and hence satisfies this criterion.

Modeling as well as dynamic modification are supported by representing workflow models in a relational database. In particular, the constituents are described in corresponding relations and tuples. Thus, *creating* a workflow model amounts to an *insertion* of tuples, while *modifying* is done by appropriate database *updates*. Moreover, combining pieces of workflow models into larger workflow models boils down to performing *retrievals*, subject to selections which test relevant assembly conditions.

Anticipated dynamic changes (cf. Section 3.) are done within a single workflow instance and, hence, should have effects only on this workflow. Consider again the workflow model shown in Figure 3.1; the modeling activity M is performed as follows: First, a number of pre-defined workflow models are displayed (subject to data flow requirements), from which the modeler can choose one to continue the incomplete workflow. The activity then sends SQL statements to the specification database to attach the chosen sub-workflow to the incomplete workflow. If an adequate workflow model is not available in the workflow library, the modeler can define a new workflow model by using the workflow specification tool. After defining the new sub-workflow, the system registers it as the continuation of the incomplete workflow, whose execution is resumed with the newly specified workflow model.

We now sketch the functionality a workflow system has to provide to support ad-hoc changes. Traditionally, for each workflow execution, there is one workflow model that is used by the system for controlling it. However, when changes to the workflow model are applied, the workflow model changes. Since there may be other workflow executions based on this workflow model,

simply changing it is not feasible. In this case, all workflow executions based on the workflow model given would be changed, something that we assume is not intended by the ad-hoc change. Therefore, a new workflow model (or a new version of the workflow model) has to be created, and the changed workflow execution is now controlled by the new workflow model. This implies that the system has to be able to use different workflow models at different times during the execution of a workflow.

The prototype allows a restricted form of dynamic change. In particular, the refinement of the sub-workflows of a given workflow may be changed. However, the change has to occur before the the start of the sub-workflow. Once the sub-workflow is started, the system does not allow changes. Future versions of the prototype will provide more flexibility for ad-hoc workflows.

5.2 System Architecture

The following design decisions have been taken, in order to achieve our main goals of flexibility and platform independence:

- *Relational representation of workflow models:* In the application domains we consider, workflows are composed of steps or collections of steps whose exact sequencing is not necessarily known in advance. We store descriptions of such steps or sub-workflows in a relational database that has relation schemas for workflow types, their constituents, I/O, data and control connectors, variables, roles, agents, etc. (see [34] for details).
- *Interpretation approach to workflow execution:* Workflow specifications are interpreted for the purpose of execution, simply by retrieving the relevant information from the workflow database. Thus, no compilation of a complete specification is done. The implication is that models can easily be changed, even while a workflow is being executed.
- *Exploitation of Java:* The workflow system has a client/server architecture, where both the workflow server and the clients are written in Java. Java byte code can be interpreted on a large variety of platforms (including the Sun Solaris and Windows95 platforms used by us).
- *Database access via JDBC:* Workflow models are currently stored in an Oracle database, and the workflow server accesses these models using a JDBC interface (Java DataBase Connectivity). Using this middleware component, different underlying relational database products can be used without changing the code.
- *Web browser as Workflow Client:* HTTP (Hyper Text Transfer Protocol) is today an industry standard for communication between Internet-connected computers. Web browsers (like Netscape Navigator or HotJava) can be used as front-ends to the HTTP protocol; these are available for a large number of platforms. We use standard Web browsers interpreting Java applets or stand-alone Java applications as workflow clients.

The design decisions and features discussed above have led to the system
architecture shown in Figure 5.1. Essentially, this is a client/server architec-
ture, where the server reads workflow models from the database, controls
the execution of workflows, and performs other important services like role
resolution. Internally, it is composed of the core component, the workflow
engine, and the database server which accesses application data stored in
the underlying database. Both components are connected to the database
by a JDBC interface, and the database contains workflow-related data (like
workflow models and role descriptions) as well as application specific data.

The workflow engine is the core part of the prototype (called "Kernel"
in Figure 5.1). When the system is started, the workflow engine reads work-
flow models from the workflow library. Since workflow models without any
incoming control connectors can be started in an ad-hoc fashion, workflow
models with this property are displayed. At any point in time, the system
may control multiple instances of a given workflow model. Hence, the system
supports concurrent executions of multiple workflows. The key functionality
of the workflow engine is to control the execution of executing workflows us-
ing information on the structure of the workflow and on the agents ready to
perform activities. In general, for each activity a role is defined, and when
the activity is about to be started, the workflow engine determines the agents
ready to perform it. This functionality is called role resolution.

We have also developed a specification tool which allows the graphical
specification of workflow models. Workflow models may be built from scratch,
or existing workflow models may be re-used, e.g., as sub-workflows in complex
workflows. On start-up, the specification tool connects to the database via
JDBC and retrieves workflow models from the database if requested by the
user. The specification tool can be invoked as a stand-alone application or it
can be invoked by a workflow application. The latter corresponds to an imple-
mentation of an anticipated dynamic change activity using the WASA pro-
totype. When the modeling activity is completed and the modified workflow
model is saved, the workflow model is effectively entered into the database.
Since workflow models are successively retrieved from the database while
workflows execute, the current workflow instance and all future instances of
that model will be effected by the dynamic modification.

Users access the workflow system using workflow clients. The basic func-
tionality of a workflow client is to inform users (agents in general) of activities
to perform. We have implemented two types of workflow clients:

- *Java Applications:* For this type, a Java interpreter has to be present on
  the client side. Since there are no restrictions on the accesses of Java ap-
  plications, implementations of workflow clients may access local data and
  may start external applications.
- *Java Applets, interpreted by standard Web browsers:* For this type, the
  presence of a standard Web browser suffices on the client side. The Java
  applet is down-loaded to the client Web browser and interpreted by it.
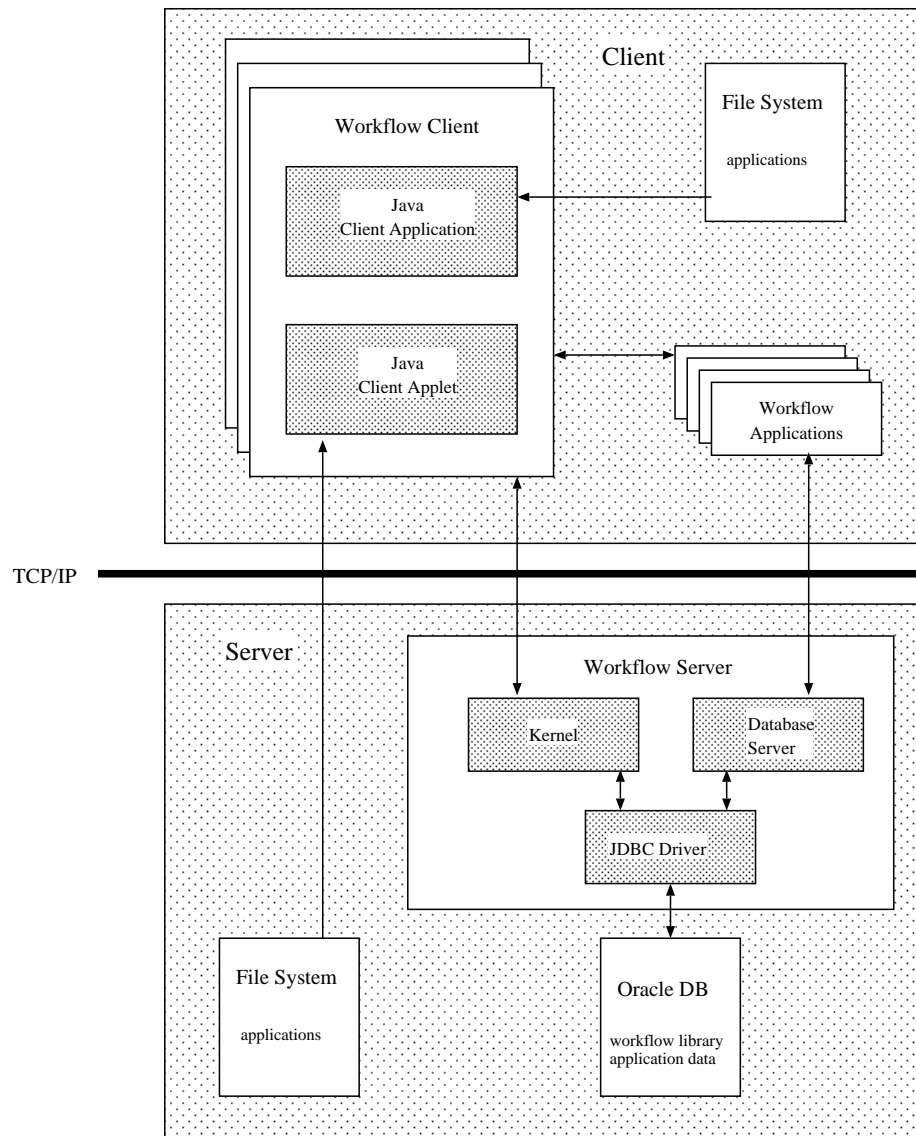
Fig. 5.1. System Architecture.

Like the other type of client, it communicates with the workflow server
using the TCP/IP protocol. However, security restrictions apply to Java
applets, e.g., access to local data and starting external applications may
not be performed by workflow clients implemented by Java applets.

Further details on the server, the two types of clients, and also on workflow
applications can be found in [31].

6. Conclusions and Future Work

In this paper we have surveyed the WASA project at the University of Muen-
ster, its goals, design decisions, and current status. The key feature is that the
WASA workflow manager is capable of supporting different dynamic change
operations which are useful for a wide range of non-traditional applications
of workflow management systems. Indeed, WASA has been designed with
scientific applications in mind, and is supposed to take care of their specific
workflow requirements. Our current prototype is a first step in realizing our
goals, and it obviously gives rise to a number of future enhancements.

Since workflow specifications, even after dynamic changes, can often be
considered as being different versions of the same underlying model, we plan
to investigate the applicability of the database version approach of [5] and its
extension into a tool for configuration management described in [6]. In the
database version approach, versioning in a (relational) database is achieved
by distinguishing versions of *entire* database instances, not just individual
relations or tuples in it. Thus, whenever a new version of some object in a
given database is created, a new version of that database as a whole comes
into being. However, the management of these versions is based on a simple
and clever naming scheme so that the overhead can be neglected, and that
issues like database consistency are easy to enforce. We expect that, since we
use a relational database for storing workflow models, a combination of the
database versioning scheme with the configuration management approach of
[6] will be fruitful.

For bringing the prototype to real-world applications, we are implement-
ing a variety of workflows from the geoprocessing domain as well as from the
area of laboratory information and management systems. We also plan to
extend our case studies to particle physics and medical applications. These
will provide a field study of the applicability of the prototype. Underneath
the workflow engine, we are exploring CORBA functionality for both sup-
port of the workflow engine and uniform access to external storage systems,
including database managers and file systems.

References

1. V. Anjur, Y. Ioannidis, M. Livny. *FROG and TURTLE: Visual Bridges Between Files and Object-Oriented Data*. In Proc. 8th International Conference on Scientific and Statistical Database Management, Sweden 1996, 76–85, IEEE Computer Society Press, Los Alamitos, CA.

2. C. B. Medeiros, G. Vossen, M. Weske. *GEO-WASA: Supporting Geoprocessing Applications using Workflow Management*. In Proc. 7th Israeli Conference on Computer Systems and Software Engineering, Herzliya, Israel 1996, 129–139, IEEE Computer Society Press, Los Alamitos, CA.

3. A. Brayner, M. Weske. *Using FlowMark^{TM} in Molecular Biology (in German)*. EMISA Forum 2/1996, 14–21.

4. C. Bussler, S. Jablonski, H. Schuster. *A New Generation of Workflow Management Systems: Beyond Taylorism with MOBILE*. ACM SIGOIS Bulletin 17 (1) 1996, 17–20.

5. W. Cellary, G. Jomier. *Consistency of Versions in Object-Oriented Databases*. In Proc. 16th International Conference on Very Large Data Bases 1990, 432–441.

6. W. Cellary, G. Vossen, G. Jomier. *Multiversion Object Constellations: A New Approach to Support a Designer's Database Work*. Engineering with Computers 10, 1994, 230–244.

7. N. Craven, D. Mahling. *A Task Basis for Projects and Workflows*. In Proc. Conference on Organizational Computing Systems (COOCS), Milpitas, CA 1995, 237–248.

8. J. Eder, H. Groiss, H. Nekvasil. A Workflow System Based on Active Databases. In. Proc. Connectivity 94, R. Oldenburg Verlag, Vienna 1994, 249–265.

9. C. Ellis, K. Keddara, G. Rozenberg. *Dynamic Change Within Workflow Systems*. In Proc. Conference on Organizational Computing Systems (COOCS), Milpitas, CA 1995, 10–22.

10. K.A. Frenkel. *The Human Genome Project and Informatics*. Communications of the ACM, 34(11):41–51, November 1991.

11. D. Georgakopoulos, M. Hornick, A. Sheth. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995.

12. N. Goodman. *An Object-Oriented DBMS War Story: Developing a Genome Mapping Database in C++*. In: W. Kim (ed.), *Modern Database Systems — The Object Model, Interoperability, and Beyond*, Addison-Wesley 1995, 216–237.

13. IBM. *IBM FlowMark: Modeling Workflow, Version 2 Release 2*. Publ. No SH-19-8241-01, 1996.

14. Y. Ioannidis (ed.). *Special Issue on Scientific Databases*. Data Engineering Bulletin 16 (1) 1993

15. Y. Ioannidis, M. Livny, S. Gupta, N. Ponnekanti. *ZOO: A Desktop Experiment Management Environment*. In Proc. 22nd International Conference on Very Large Data Bases 1996, 274–285.

16. S. Jablonski. *MOBILE: A Modular Workflow Model and Architecture*. In Proc. 4th International Working Conference on Dynamic Modeling and Information Systems, The Netherlands 1994.

17. N. Kamel, T. Song, M. Kamel. *An Approach for Building an Integrated Environment for Molecular Biology Databases*. Distributed and Parallel Databases 1, 1993, 303–327.

18. F. Leymann, W. Alterhuber. *Managing Business Processes as an Information Resource*. IBM Systems Journal 33, 1994, 326–347.

19. R. McClatchey, N. Baker, W. Harris, J.-M. Le Goff, Z. Kovacs, F. Estrella, A. Bazan, T. Le Flour. *Version Management in a Distributed Workflow Application.* Proc. 8th International Workshop on Database and Expert Systems Applications 1997, Toulouse, IEEE Computer Society Press, 10–15.
20. J. Meidanis, G. Vossen, M. Weske. *Using Workflow Management in DNA Sequencing.* In Proc. 1st IFCIS International Conference on Cooperative Information Systems (CoopIS), Brüssel, Belgium 1996, 114–123, IEEE Computer Society Press, Los Alamitos, CA.
21. E. Mesrobian, R. Muntz, E. Shek, S. Nittel, M. LaRouche, M. Kriguer. *OASIS: An Open Architecture Scientific Information Systems.* In Proc. 6th International Workshop on Research Issues in Data Engineering 1996, 107–116.
22. Object Management Group and X/Open. *The Common Object Request Broker: Architecture and Specification.* OMG Document 91-12-1, 1991.
23. R. Orfali, D. Harkey, J. Edwards. *The Essential Distributed Objects Survival Guide.* John Wiley & Sons 1996.
24. M. Reichert, P. Dadam *A Framework for Dynamic Changes in Workflow Management Systems.* Proc. 8th International Workshop on Database and Expert Systems Applications 1997, Toulouse, IEEE Computer Society Press, 42–48.
25. T. Reuß, G. Vossen, M. Weske. *Modeling Samples Processing in Laboratory Environments as Scientific Workflows.* Proc. 8th International Workshop on Database and Expert Systems Applications 1997, Toulouse, IEEE Computer Society Press, 49–55.
26. B. Rieche, K. Dittrich. *A Federated DBMS-based Integrated Environment for Molecular Biology.* In Proc. 7th Int. Working Conf. on Scientific and Statistical Database Management 1994, 118–127.
27. A. Sheth, K. Kochut, J. Miller, D. Worah, S. Das, C. Lin. *Supporting State-wide Immunization Tracking using Multi-Paradigm Workflow Technology.* Technical Report UGA-CS-TR-96-001, University of Geogia, February 1996.
28. L. Stein, S. Rozen, N. Goodman. *Managing Laboratory Workflow with LabBase.* In Proc. 1994 Conference on Computers in Medicine.
29. K.D. Swenson and K. Irwin. *Workflow Technology: Tradeoffs for Business Process Re-engineering.* In COOCS'95, pages 22–29, Milpitas, CA, 1995.
30. G. Vossen, J. Becker. (editors) *Business Process Modeling and Workflow Management: Models, Methods, Tools* (in German). International Thomson Publishing, Bonn, Germany, 1996.
31. G. Vossen, M. Weske, G. Wittkowski: *Towards Flexible Workflow Management for Scientific Applications*; manuscript, University of Muenster, 1997.
32. M. Weske. *Flexible Modeling and Execution of Workflow Activities.* To appear in Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31), 1998.
33. M. Weske, G. Vossen, C. B. Medeiros. *Scientific Workflow Management: WASA Architecture and Applications.* Fachbericht Angewandte Mathematik und Informatik 03/96-I, Universität Münster, 1996.
34. G. Wittkowski. *Design and Implementation of a Workflow System in Java (in German).* Diploma Thesis, Universität Münster, 1996.
35. Wodtke D., Weissenfels J., Weikum G., Kotz Dittrich A.: The Mentor Project: Steps Towards Enterprise-Wide Workflow Management. In Proc. 12th IEEE International Conference on Data Engineering (1996), 556–565