WASA: A Workflow-Based Architecture to Support Scientific Database Applications*

Claudia Bauzer Medeiros†
University of Campinas
Brazil

Gottfried Vossen, Mathias Weske‡
University of Muenster
Germany

January 1995

Abstract

Scientific applications which involve the use of databases are of emerging interest. However, computer-based environments which support such applications are still in their infancy. The overall goal of the WASA project is to provide such an environment, and to do so with the specific domains of geosciences and biocomputing in mind. This paper introduces the architecture of WASA, explains its design decisions, describes how its various components interact, and how it can be utilized in the target applications. One important design decision has been to reuse existing database technology as far as possible, but to integrate this with the emerging paradigm of workflow management. As a result, WASA has a workflow manager as its central component; this tool manages information as well as other tools associated with a given environment, some of which are domain-independent, while others have to be adapted whenever the system is used in a new application area.

1 Introduction

In recent years there has been an increasing interest in scientific databases or, more generally, in using database technology in scientific environments, as a result of the need to support scientists (e.g., in the natural sciences) in handling large volumes of complex-structured data in an adequate and efficient way [3, 9, 20, 19]. However, most efforts have so far focused on the *management* of scientific data, i.e., on providing scientists

*This work was in part supported by the German Ministry of Science and Technology (BMFT) and by CNPq Brazil, within a bilateral cooperation on Database Technology and Knowledge-Based Systems, under Grant No. 30.3.I1A.6.B

†Departamento de Ciencia da Computacao, University of Campinas (UNICAMP), 13081 Campinas S.P., Brazil. E-mail cmbm@dcc.unicamp.br

‡Institut fuer Wirtschaftsinformatik, University of Muenster, Grevener Strasse 91, D-48159 Muenster, Germany. E-mail {vossen,weske}@uni-muenster.de

with techniques and tools to store, retrieve, manipulate, and manage scientific data; in particular, many domain-specific proposals have been made and tools developed which support the daily work of scientists in a variety of scientific domains [7]. This paper takes a broader perspective on the issues, by proposing an *environment* for supporting scientific database applications, in which the emerging paradigm of *workflow management* plays a central and integrating role.

While pure data handling and management issues are crucial in the applications in question, there are many other aspects which genuine and dedicated database support for scientific environments has to capture; these include, for example, mechanisms for cooperative work, distributed and parallel data management, or the management of the scientific experiments. Hence, *environments* by which the everyday work of a scientist can adequately be supported need to go way beyond pure data management. Indeed, it cannot even be generally assumed that a scientist whose work deals with databases is a computer specialist, or is willing to study the details of computer support. As a result, we feel that such environments need to be constructed in a way that takes these various issues into account.

The goal of this paper is to describe a proposal for building such an environment. In particular, we will describe the architecture of *WASA* (**W**orkflow-based **A**rchitecture to support **S**cientific **A**pplications), a system intended to integrate database technology with tools existing in a scientific environment, and to ultimately support the management of scientific experiments. One important design decision of WASA has been to re-use existing database technology as far as possible, but to integrate this with the emerging paradigm of workflow management. As a result, WASA provides scientists with an environment which helps in planning, organizing, conducting, evaluating, documenting, and disseminating results of scientific experiments. The WASA architecture is described here primarily from a database perspective, since we assume that data generated in experiments and manipulated by analysis procedures will be stored in some type of database; however, we are not concerned with accesses to such databases, but assume that this will be handled by some kind of standardized interface [15, 16].

We mention that, while database management in scientific applications has been considered important for some time already, integrated support for scientific environments and workplaces has rarely been studied so far. Indeed, work related to ours includes experiment management tools [8, 24], extended semantic data as well as object models [11], tools related to such models [12, 21, 13], systems for developing and integrating scientific and engineering software [23, 17, 20], the provision of class hierarchies [19, 25], and architectures for cooperation using, and interchange of, scientific data stemming from experiments [6, 1]. However, little thought has been given to providing a general "umbrella" in which such tools coexist with tools related to the application domain in question. An exception is the XBio system [9], which makes intensive use of federated database technology to provide an integrated user environment. In XBio, each user has a federated view on the application program suite of interest. In contrast, WASA tries to hide an eventually existing database federation through the incorporation of an object broker interface based on common standards [15, 16], which may even involve open transaction processing as in [22], and it emphasizes the work aspect of scientists through

the use of a workflow manager. It should be mentioned that the workflow manager subsumes cooperation, which is also relevant to scientific experiments; hence cooperation can be achieved within WASA, but there is no component besides the workflow manager to provide a cooperation environment.

The remainder of this paper is organized as follows: In Section 2 we elaborate on our perception of how scientists conduct experimental work, and motivate why workflow management appears to be an appropriate paradigm to support scientific applications. The WASA architecture is then described in Section 3. Section 4 provides some initial examples of how this proposal can be used in different domains; in particular, we look at fragment assembly in molecular biology, and at an ecological experiment to be conducted in the area of geosciences. Finally, we present some conclusions and directions for future work.

2 Preliminaries

In this section we give a rough overview of our perception of how scientists work; this view is the result of a number of discussions we had with scientists in the bio-science and in the geoscience domains. After that, we describe an abstract model of scientific work, which tries to reflect our findings. While patterns of scientific work vary from one domain to the next and can hence not be generalized, several common work patterns can be identified; our results are applicable to scientific environments with these patterns. We then collect the key functionality features needed in a scientific environment, and motivate why workflow management seems to be an appropriate paradigm to integrate these features.

2.1 A Model of Scientific Work

Traditionally, scientific work is centered around experiments. As an example, consider the development of some novel medication for a disease. The chemical lab of a pharmaceutical company will first try to identify a number of substances that may help to cure the disease; these will be designed according to experience, legal requirements, cost aspects, and such that, for example, undesired side-effects are minimized. In order to do so, the lab will conduct a huge number of experiments, eventually move to animal tests, and later experiment with humans in a hospital environment. If successful, the new medication will have to go through intensive approval procedures and eventually reach the medication market.

From a database point of view, the scientific experiments which have to be carried out during such a development process are typically composed of a series of *steps* or *tasks*, which are intertwined according to some control sequence (e.g, loops, conditional branches, and parallel execution). Each such step receives some input information and produces output information. This information is manyfold. It may appear in forms such as stored data (e.g., files), data received from external devices (e.g., sensors), description of devices to be used on the execution of a step (e.g., equipment specification),

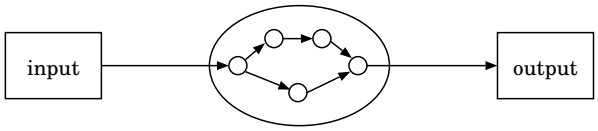

Figure 1: General View of an Experiment.

conditions concerning step execution (e.g., preconditions, interrupt situations), consistency constraints, scheduling information, and others.

For the purpose of this paper, we assume that all such information is stored in data sets, which are managed by some sort of database management system (DBMS). By *data set* we mean any set of data that is scientifically meaningful. Data sets may be classified according to their use in experiments. Each step of an experiment is based on a number of *input data sets*. It can only be conducted provided all input data sets are available to that step. When the step is conducted, it produces some *output data set*, which may be subject to further analysis; in addition, the output data set of one step may be used as the input to a subsequent step.

The execution of individual steps may be restricted to obey some order. Two steps are ordered if, for instance, the output of one step is required as the input of another. Other reasons for imposing an order on steps may be implied by their resource requirements. Notice that execution ordering of steps is a simple form of *constraints*. There may be other such constraints, e.g., causal constraints ("Step 3 is executed if and only if Step 2 is not executed"), or constraints based on result values ("Step 2 is executed if and only if Step 1 has returned a certain value"). (If there are no constraints between two tasks, they can in principle be executed concurrently.) So in general, there are constraints of tasks that are implied by the application, and there are constraints implied by external conditions. Furthermore, control structures used in programming languages can also be used to specify scientific experiments. Finally, individual steps are not necessarily atomic, and may be divided into sub-tasks, which can again be subject to control conditions.

As a result, a global abstract view of a *scientific experiment* is that it consists of one or more steps with input and output data; this view is depicted in Fig. 1. Notice that the core of an experiment is composed of a number of steps, some of which may be executed in parallel. In graphical terms, we will show an edge from a step s to a step t whenever the output of s will be the input of t. Notice that the notion of *step* as a "black box" with input and output is quite general; it allows a step to correspond to any activity within a scientific environment (e.g., number crunching, data editing, result visualization, etc.).

We finally mention that an important scientific task is to perform an *analysis*, which can itself be perceived as an experiment as just described. Analysis procedures are techniques that transform or derive new data from existing information. Examples of such procedures are statistical methods (e.g., in medicine), pattern matching procedures (e.g., in molecular biology), or structure prediction and analysis methods (e.g., in structural biology).

2.2 On the Functionality of Scientific Environments

We now move from conceptual to physical issues and describe the key functionality which must be provided by any scientific environment in order to be useful; this list is incomplete (for example, it does not emphasize cooperation), and has to be extended based on the specific requirements of a given application domain.

- **Reproduceability and dissemination:**
  Scientific results must be reproducible and disseminated. To ensure this, experiments must be well documented, in a way that will allow other scientists to reconduct the experiment and confirm the results. Results may be used as a basis for future research, either by the scientists who generated the data, or by others.

  Documentation is also necessary for dissemination purposes. Result dissemination is crucial for the successful utilization of scientific results.

- **Experiment Design:**
  Scientists should be assisted in the design of experiments. To investigate a hypothesis or to show some new property of a subject, scientists design new experiments. Experiments are built from scratch if no previous experience is available, or they may re-use and combine steps of previous experiments. In practice, re-using steps of previous experiments is a common choice, since certain parts of experiments may be relevant for more than one experiment.

- **Controlled Conduction of Experiments:**
  Often, experiments consist of a number of steps with complex constraints. These steps may be executed in a distributed environment, i.e., multiple steps may run in parallel using different devices. Often, certain steps are not successful and have to be re-executed, for instance due to a contamination of chemical agents in a chemistry experiment.

If data used in and produced by experiments is kept in a database, the system managing this database, i.e., the underlying DBMS, should provide additional capabilities that are relevant to experiments; these capabilities include general database functionality such as secondary storage management, multi-user processing, fault tolerance, and efficient access to huge volumes of data. However, equally important seems to be advanced functionality such as *temporal data management, versioning*, and *activeness*; since neither of these are commonly found in DBMSs which are currently available in the marketplace, we have decided to make them part of our architecture, as will be described in the next section.

2.3 Scientific Workflows

The view that scientists typically perform experiments, and that experiments can be considered as ordered collections of steps acting on data and involving a variety of distinct activities, motivates the exploitation of the emerging paradigm of *workflows* and *workflow management* in this context. in general, a *workflow* denotes the controlled

execution of multiple tasks in an environment of distributed processing elements [18]. As we have discussed, specifying, monitoring and controlling the conduction of complex experiments and analyses are important tasks of a scientific environment. On the other hand, controlling the execution of complex activities in a distributed environment is generally recognized as the main activity of a workflow management system.

We now explain important properties of scientific workflows, by comparing them to traditional, i.e., business workflows [10]. In the latter, a workflow is specified once and then executed numerous times in a routine fashion. However, scientific workflows are generaa not executed routinely; there may even be applications in which each workflow is executed only once. In business applications, the main motivation for introducing workflow management is the desire to organize (actually "re-organize" or "re-engineer") work in, say, a company to enhance its efficiency. The motivation for workflow mangement in scientific applications, however, is less to enhance efficiency, but to control experiments, and to make available to scientific users the information on how experiments were conducted. What both domains have in common is the need to organize and control process executions.

Scientific workflow management systems, in a sense, generalize traditional workflow managers, in a similar way workflow managers generalize multidatabase transaction managers [18]. Both types of workflow managers are responsible for supporting workflow design and concurrent execution of multiple workflows in a distributed environment. Hence, standard facilities provided by existing workflow management systems regarding definition, execution, and user control will be important for scientific workflow managers as well. In addition, new functionality will be needed with respect to recording workflow executions for later re-use, or to tracing workflow executions. Indeed, operations a user performs on given data must be recorded and stored together with some form of pointers to the data that was manipulated. In doing so, full documentation on the operations executed can be annotated to data sets, ensuring reproducibility of results. Furthermore, these workflows that are produced by working with the system should recursively be stored by a documentation tool, to make all this information available to later experiments. It is interesting to note that the scientist may benefit not only from stored successful workflows, but also from workflows that turned out to be unsuccessful. The latter could be achieved by warning the user that a step taken proved unsuccessful in previous experiments.

Another difference between traditional workflows and scientific workflows is that in the former there is a specification phase during which the relevant workflows are specified. Once this phase is finished and execution is controlled by the system, workflows remain vastly invariant over time. On the other hand, in a scientific setting a specification may change even while a workflow is being executed. In addition, users in a scientific environment will typically specify their workflows themselves, while a system administrator (or a business-process specialist) is commonly responsible for this task in traditional environments.

These differences must have implications on the structure and organization of specifications of scientific workflows. Since existing workflows may be re-used for designing new ones, there are generally relationships between distinct workflow specifications, which

can be described in the form of a directed graph in a way similar to what is commonly used when managing versions of data items. It follows that a scientific workflow manager should be capable of managing *versions of workflows*, which can even be accessed via specific operations (on workflows). Once such a "meta-level" for acting upon and reasoning about workflows is available, it is natural to also allow that workflow specifications may be incomplete, i.e., that workflows are only partially specified. This is motivated by the observation that it is often that case that, prior to the execution of an experiment, the complete workflow corresponding to this experiment is unknown. On the other hand, the result values of certain steps may indicate how to proceed. Thus, while an initial part of the workflow has been defined before its execution was launched, its remainder will only become known while it is under execution.

In summary, the goals of scientific workflow management include

- allowing the specification of eventually incomplete workflows,
- ensuring the reproducibility of results,
- controlling the execution of complex experiments,
- support of dynamic modifications of pre-defined workflows,
- allow re-use of existing workflows.

In the following section, we will elaborate on how the architecture of the WASA system is intended to achieve these goals.

3 Architecture of the WASA System

In this section we describe the architecture of WASA. As we said in the Introduction, its overall goal is to provide an integrated environment for scientific work, which makes intensive use of database technology, and which is based on the paradigm of workflow management, suitably adopted to the scientific domain. In the system architecture, the following four layers can be distinguished:

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

We describe these layers in detail next. Figure 2 gives an overview of the WASA architecture.

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 system.

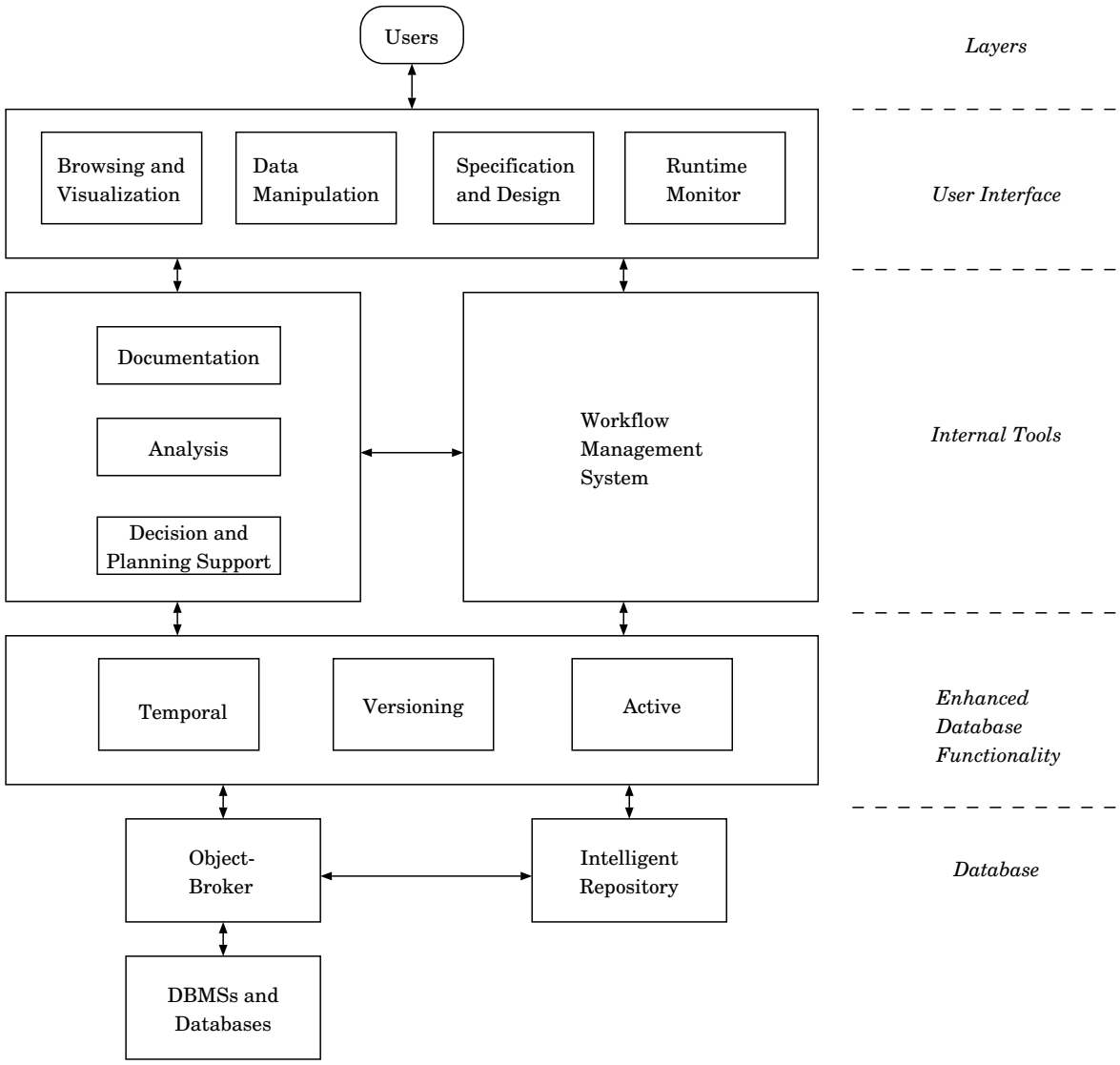

Figure 2: The Architecture of WASA.

3.1 The WASA Layers

User Interface Layer

The *User Interface Layer* is responsible for communication between users and 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 as 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). One such data manipulation box could consist of a database query language processor.

The *Visualization* facility 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 be visualized.

Finally, the *Runtime Monitor* allows users to execute previously defined workflows, and to monitor and control their executions.

Internal Tool Layer

The *Internal Tool Layer* consists of the workflow management system and of a set of auxiliary managers to support experiment specification, documentation, and execution. The *Workflow Management System* is the core component of the WASA system. It provides the relevant functionality for scientific workflows as described in Section 2. 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 of how to continue a given experiment would also be a task of this component, i.e., which step to take next in a given experiment. 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.

Enhanced Database Functionality Layer

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 right now to build scientific applications on legacy database systems such as relational ones. As a consequence, the 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:

The *Temporal* management facility allows users to maintain historical information, e.g., of meaningful events, of past experiments, as well as the history of how experiments were used to make up other experiments. It also allows managing activities in the future (such as experiment planning). The *Active* management mechanism supports controlling the execution of a given task or workflow, as well as other needs such as integrity maintenance during workflow execution. Due to this mechanism, different workflow executions may be triggered depending on the outcome of a given experiment. Finally, *Version* management is responsible for dealing with versions of data items as well as with versions of workflows.

In order to make this layer portable and independent of pre-existing databases in particular application domains, the WASA system is intended to provide access to individual databases through a standardized interface. To this end, it is reasonable to assume that underneath WASA typically is a federation of multiple databases with a common interface such as an object request broker [15, 16] 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.

Database Layer

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:

- Application- and domain-specific data, which is supposed to be stored in a number of databases whose details are hidden from the system through the broker. This includes data that may appear in all domains, like text data (persons, labs, addresses), references to bibliography data, and other data resources. General data also includes internal data like data on experiments, on devices, and on analysis procedures.

- An intelligent data repository. This is a reserved and 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 (e.g., structures to manage the temporal dimension).

3.2 Specification and Running of Experiments

We now describe how the different building blocks and layers of WASA interact. We will do so here without reference to a particular domain of application; the next section will then be devoted to sample domains WASA is intended to support. Generally, the workflow management system plays the central role, and is supported by the internal tools and enhanced database functionality.

In order to understand the interplay among the various parts and components of WASA, we consider a sample user who first specifies a new experiment and then executes it. Initially, the user interacts with WASA using the specification and design facility. Through it, the user can start designing a new workflow, which may even begin with an inspection of previously designed workflows. The selected workflow may need to use some analysis functions (which are therefore offered to the user by means of combined access to the workflow management system and the analysis manager). At some point, the user may want to re-use parts of previous experiments. In this case, the user will interact with the browsing and visualization box which, through an interaction with documentation and the workflow management system, will provide the desired data. In case of doubts as to which parts to re-use, the user may be assisted by the decision-support and planning manager.

Throughout a specification phase, user activities are recorded by the documentation manager. Since the specification of some step may, at execution time, trigger the underlying active database management facility, the documentation manager should be capable of indicating which rules may eventually be fired during an execution of the workflow under specification. Furthermore, the creation of different workflow patterns will normally activate the versioning mechanism; finally, the temporal management facility may be needed in order to support the use of historical data, or to record modifications made to data. In all these cases, data about and used by the workflow, documentation about the user activity as well as any control data needed to run this workflow are recorded in the underlying databases as appropriate or applicable, and in the WASA repository.

When the user considers a specification complete (or at least ready for being executed), the runtime manager may be invoked to execute the new experiment, and the results will be seen through the browsing and visualization facility. A run of the experiment will itself trigger the documentation and analysis facilities, as well as different functions to be performed by the database management layer.

The information on workflows and their respective successes are not stored by the workflow management system, but by the documentation manager. All data needed is sent from the workflow manager to the documentation manager, which is also responsible for retrieving information on previous workflows. Therefore, documentary annotations must be affixed to each data set generated, specifying how that data set was obtained. These annotations may include specification of the input data sets, the steps taken, and other information.

Analysis procedures that are implemented within the system and information on the usable external analysis procedures are controlled by the analysis manager. Whenever an external analysis is to be used, information on its semantics must be provided (e.g., data formats, semantics of transformation, annotational information, person responsible for implementation, algorithm used, references, etc.). Additionally, all information necessary to start the procedure must be available, since automatically starting programs is an important functionality of workflow management.

We mention that there is an ongoing controversy of where a handling of analysis procedures should take place. The options are to do this either within the system or outside the system. Implementing them within the system allows to monitor all transformations applied to a given data set; however, this would be done at the expense of a necessary re-implementation once the system is transferred to a new application domain. Conversely, if analysis procedures are stored externally, data has to be exported from an underlying database, and newly created data has to be imported back into these databases. In this case, the system loses control over what has happened to the data during analysis, but this seems acceptable from a flexibility point of view.

The WASA architecture tries to combine the two approaches. Analysis procedures can be implemented either outside or inside the system. The point is that each analysis run on a given set of data is controlled by the system. Instead of exporting and importing data, the semantics of each analysis procedure (input format, transformation semantics, output data format) must be defined. Analysis procedures are triggered by the system: Since the semantics of the procedures are known and the system knows on which data sets they were run, the system has complete information on the transformations that were applied to a given data set. In that case the requirement of reproducibility can be guaranteed, now combined with re-using existing analysis procedures and avoiding expensive re-implementing analysis procedures within the system.

4 Sample Applications

In this section, we provide two specific examples of applications in which we consider a use of WASA relevant and fruitful. As will be seen, these two examples have a number of distinct characteristics, and hence exhibit a wide spectrum of scientific environments in which WASA appears to be applicable.

4.1 Fragment Assembly in Molecular Biology

Our first example is from the area of molecular biology and considers *fragment assembly*. To this end, recall that all genetic information of organisms is stored in nucleotide sequences. One form is desoxyribonucleic 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 [2, 4]. Today, short sequences of DNA ($\leq$ 500 base pairs) can be generated semi-automatically, meaning that biochemical agents are fed into a device which produces the respective base sequence [14].

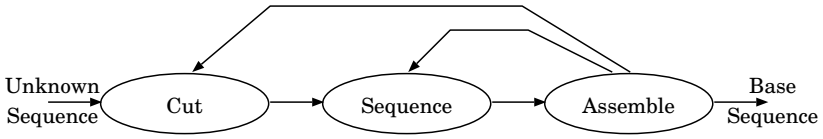

Figure 3: Fragment Assembly.

In general, scientific experiments are error-prone. In the context of DNA sequencing, contamination of chemical agents or device inaccuracies are among the sources of errors. Hence, 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.

Traditionally, information on the validation procedures available and on the conduction of analysis steps have to be maintained by the scientist, e.g., using paper notebooks. However, WASA can support this process as follows:

1. Using the browsing facility of WASA (which accesses the analysis manager), the user can access information on the validation procedures that are available.

2. Using the data manipulation interface, a validation procedure is executed. This includes feeding the raw data set into the procedure, eventually performing format transformation operations, and generating an output data set.

3. Using the information of the semantics of that procedure (by accessing the analysis manager), the documentation manager appends annotational information to the generated data set, thereby ensuring complete documentation of the newly generated data set.

Since direct sequencing is possible only for relatively small sequences, in order to sequence larger stretches of DNA (e.g., 10K base pairs), a divide-and-conquer strategy has to be used. Such strategies are known as *fragment assembly* and 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.

This sequence of steps describes an ideal situation. In reality, however, the fragment assembly process does not consist of a linear sequence of experiments and analyses. Instead, there may be branches as well as loops. A more realistic description of the situation is hence as illustrated in Fig 3.

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.

In this context, workflow management appears to be an important and appropriate paradigm. Indeed, the workflow manager of WASA will help in controlling and managing the conduction of complex experiments. In particular, information on the progress of each sequencing experiment must be available to the user in order to react to situations that would otherwise hamper the experiment's success, e.g., to put more resources into slowly running experiments, or to avoid redundant work. In more detail, the workflow management component of WASA can support the fragment assembly process as follows:

1. Control and monitor the execution of fragment sequencing experiments that are conducted in parallel,

2. react to situations in which the progress of the overall experiment depends on the success of a sub-experiment (like sequencing a fragment),

3. store information on the experiments executed in order to ensure reproducibility of results.

Furthermore, using the documentation facility of WASA, information on the parameters of assembly steps are available to resolve conflicts in data sets. In particular, an assembly of two fragments is of higher quality if the overlap is longer, and the percentage of matches higher. In a later step, two data sets may have conflicting information. Then it would be important to have information on the quality of the data sets. If one data set is of high quality and the other of rather low quality, the latter will be discarded. For this reason, maintaining control on the analysis taken and a provision of full information on the history of data sets can improve the efficiency of the fragment assembly process.

4.2 Experimental Ecology

Our second example of how the features of WASA can be exploited in scientific applications considers an ecological experiment, namely the impact of vegetation destruction on hummingbird populations. To this end, we mention that hummingbirds primarily feed from flowers. In some ecosystems live different species of hummingbirds, each with distinct feeding habits and flight characteristics. As they feed and fly, hummingbirds help pollinize a given area, thereby guaranteeing the reproduction of the flowers they feed from, and thus maintaining the ecological balance.

The experiment we describe for illustration purposes consists in determining the impact of destroying certain parts of a given forest, as far as the hummingbird population is concerned. The result is a set of maps with alternative scenarios, indicating possible

effects of devastation for different simulation conditions. Most of the input needed (e.g., hummingbirds sighted per forest location) needs actual ground survey. Thus, a considerable part of experiment description involves specification of data collection and validation procedures. Data collection for this type of experiment typically requires (cooperative) field-team work during some time, with comparison of data collected in order to guarantee a certain level of quality.

In terms of WASA, this requires specifying and running a workflow with the following input data: forest area to be studied; its vegetation cover, including flower species; hummingbird species of the forest, and their feeding and flying characteristics. The workflow specification must include the following steps (see Figure 4):

- Ground data collection for vegetation cover and bird distribution:

  Ground data collection may be conducted by several teams in parallel, each of which is given a specific sub-area to cover. These teams must exchange control information, which will allow cross-checking of results, as well as defining adequate values across sub-areas.

  Vegetation cover may partially be obtained by an analysis of satellite images or aerophotogrammetry. Finer grain details (e.g., flower distribution) requires ground survey. This data will be added to the remote sensing information, with spatial indication of predominant species. Biological information on hummingbirds comes from existing textual files and databases. Spatial distribution will again be provided by on-site data collection (number and species sighted at a given site and time).

- Intermediate data analysis:

  This will involve two types of analysis functions:

  1. Production of intermediate vegetation and bird maps.
     This can be done in parallel, by means of automated analysis functions. The resulting maps will show the forest subdivided into sets of predominant flower and bird species.
  2. Quality checking.
     The intermediate maps must be checked against each other. Hummingbird and flower species must be mutually consistent for each sub-area studied. Discrepancies may arise either from incorrect hypotheses on intermediate map generation or in ground survey. Inconsistencies may require backtracking.

- Output production:

  If input data is considered acceptable, the final analysis of data can begin, in which the intermediate maps generated are cross-checked against the areas intended for devastation. Here, several different scenarios have to be generated, to account for other factors (e.g., extension and type of destruction).

The interaction of a user with WASA will occur both at the experiment specification and at the execution level:

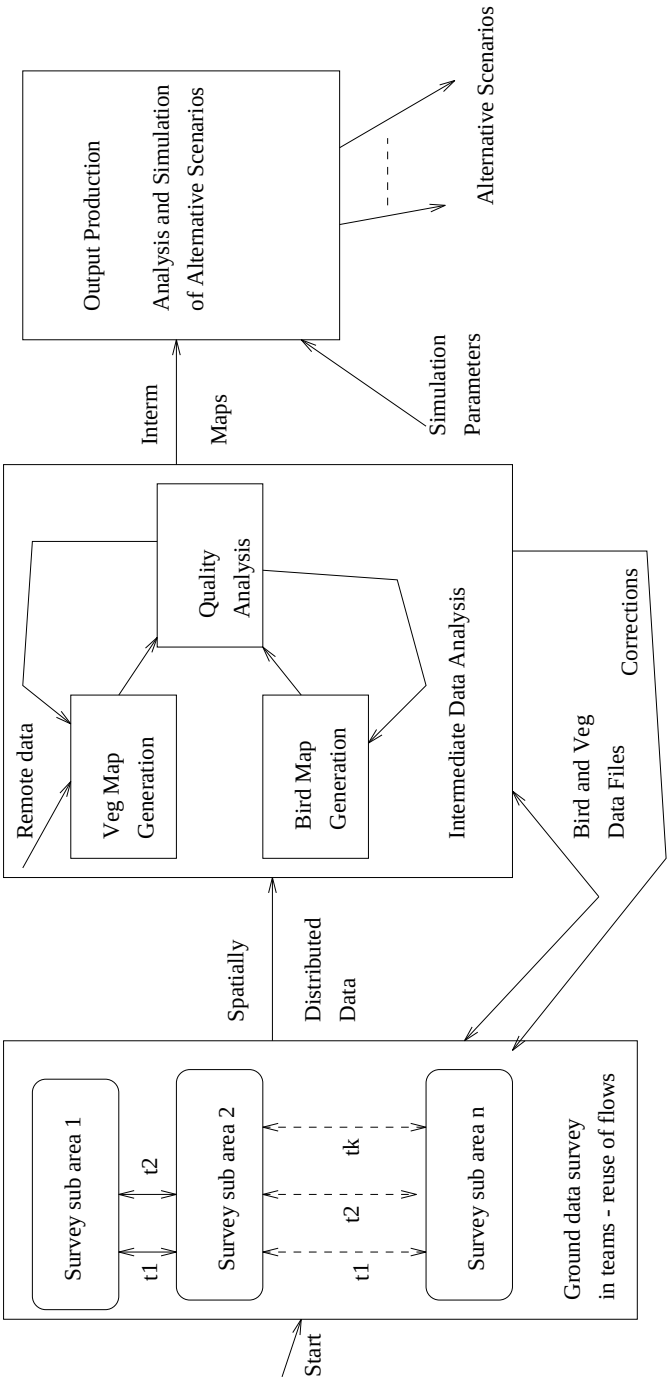

Figure 4: An Ecological Experiment.

1. Experiment (workflow) specification:
The user will specify the experiment steps using the specification and design facilities. At the same time, document management will store data about the experiment being designed (data needed for the experiment as well as design history).

Ground data collection is usually based on a few well-known procedures. Thus, the user may want to re-use specifications of previous and similar experiments, changing them only in what concerns the area of analysis, and factual data on birds and vegetation species. In selecting the proper ground survey experimental procedures to be re-used, the user is assisted by the decision and planning support system, in interaction with the document management facility.

If several teams are to perform ground surveys in parallel, the user will specify the relevant sub-workflows as often as necessary (as shown in the first box of Figure 4), and indicate that cross-checking among teams be conducted at some time periods. This requires interaction of the workflow management system and the active database functionality.

The other steps (intermediate analysis and computing alternative scenarios) are performed through analysis procedures. Here, the user will define or re-use the appropriate analysis functions, being supported in this activity by the analysis module in conjunction with the decision support module, in order to obtain the alternative output scenarios.

2. Experiment monitoring:
Here, the runtime monitor interacts with the workflow management system. When the workflow is started, the latter provides ground survey teams with instructions on conducting the survey. The active database component will ensure that the groups will cross-check their data during their survey activities. Data will be recorded by means of interaction with the data manipulation module, together with information about when it was collected (to be managed by the temporal database functionality).

Next, intermediate map generation will be performed. In particular, the workflow management system will interact with the analysis module, and access the databases for collected and repository data (e.g., for constraints on workflow execution). Database accesses will take advantage of both the active and the temporal facilities, in order to conduct the appropriate steps in the analysis. Some data discrepancies will be corrected automatically (via the active facility), whereas others will need user intervention. In both cases, the intermediate maps will be made available to the user by means of interaction of the runtime monitor with the browsing and visualization module.

Finally, the different output maps will be stored as different versions, to be managed by the version database functionality. These versions will be made available to the user by means of the browsing and visualization module. Each version will be associated with an annotated history of how it was produced (data and functions used). This history, provided by interaction of the documentation facility and the temporal data management, is necessary for future interpretations of output results, as well as for guaranteeing their reproducibility.

5 Conclusions and Future Work

In this paper, we have outlined an architectural framework for supporting scientific applications in which huge amounts of data have to be managed in an efficient and application-specific way. The key features of the WASA system include that

1. it is constructed according to distinct layers of abstraction and functionality, which clearly separate, but also include the core components upon which work in scientific environments relies,

2. it can be build atop arbitrary existing types of databases, since it assumes a common interface to these (via an object broker component) which uses emerging standards for interoperability on databases,

3. its operation crucially centers around the paradigm of workflows and workflow management; indeed, the general perception is to view all activities arising in a scientific environment, in particular the steps and tasks in an experiment, as part of some form of workflow, so that a support system needs appropriate functionality for their specification, control, and execution.

Most research on scientific databases, or on databases used in scientific environments, has so far centered around the support of scientific data management. In this paper we have added a new dimension to database support for scientific environments, by proposing the WASA open architecture to support the development of scientific experiments. WASA allow users (expert scientists) to specify new experiments and to reproduce and re-use past experiments or parts thereof. The main contributions of this architecture are:

- Separation of experiment specification, control, and execution from experiment control and documentation;

- support for dynamic execution of tasks, by combining active and temporal database facilities;

- support for experiment re-usability and reproducibility, by means of the documentation and versioning facilities;

- scientific domain independence: all necessary functions for a given experimental domain can be stored in databases and function libraries, accessed by specific modules. The repository will then be used by the environment to provide data for execution of the tools of WASA.

- Support for dissemination of experimental procedures results, both via the documentation facility and by providing the facility of keeping relevant data on the repository.

As we have tried to indicate by looking at two sample applications, the WASA architecture is general enough to be adapted to a variety of scientific environments; it can even be used in other areas, like CAD, or database design. To adapt WASA to a new domain,

what is essentially necessary is to provide the appropriate decision support rules and design and control data to the data repository.

Future work on WASA will be done in various directions. First, we plan to build an experimental implementation atop a commercial object-oriented DBMS, in order to get feedback on the feasibility of the architecture. Second, we are currently developing a model of workflows in which the aspects of partiality and versioning play a central role; this model will form the basis of WASA's specification component. Other open issues not analyzed in this paper include, for example, integrity management during concurrent workflow execution, the functionality of the active system components, and extended transaction support for scientific applications.

References

[1] M. Armstrong, P. Densham. *Toward the development of a conceptual framework for GIS-based collaborative spatial decision-making.* Proc. 2nd ACM Workshop on Advances in Geographic Information Systems 1994.

[2] C. DeLisi. *The Human Genome Project.* American Scientist 76, 1988, 488–493.

[3] J.C. French, A.K. Jones, J.L. Pfaltz. *Summary of the Final Report of the NSF Workshop on Scientific Database Management.* ACM SIGMOD Record, 19 (4) 1990, 32–40.

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

[5] 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.

[6] A. Hemerly, A. Furtado, M. Casanova. *Towards Cooperativeness in Geographic Databases.* Proc. 4th DEXA 1993, 373–376.

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

[8] Y. Ioannidis, M. Livny, E. Haber, R. Miller, O. Tsatalos, J. Wiener. *Desktop Experiment Management.* Technical Report, CS Dept. University of Wisconsin, Madison, WI, 1994.

[9] 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.

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

[11] V.M. Markovitz. *The Object Protocol Model.* Technical Report, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, CA, 1994.

[12] V.M. Markovitz, W Fang. *SDT 4.1. Reference Manual.* Technical Report LBL-27843, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, CA, 1991.

[13] V.M. Markovitz, A. Shoshani. *Query Specification and Translation Tools. Design Document 1.7.* Technical Report LBL-31155, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, CA, 1991.

[14] J. Meidanis. *Rethinking the DNA Fragment Assembly Problem.* Technical Report DCC-23/93, UNICAMP, University of Campinas, Brazil, 1993.

[15] Object Management Group and X/Open. *The Common Object Request Broker: Architecture and Specification.* OMG Document 91-12-1, 1991.

[16] Object Management Group. *Object Management Architecture Guide.* Revision 2.0, OMG Document 92-11-1, John Wiley & Sons 1992.

[17] N. Pissinou, K. Makki, E. Park. *Towards the Design and Development of a New Architecture for Geographic Information Systems.* Proc. 2nd International Conference on Information and Knowledge Management 1993, 565–573.

[18] M. Rusinkiewicz, A. Sheth. *Specification and Execution of Transactional Workflows.* In: W. Kim (ed.), *Modern Database Systems — The Object Model, Interoperability, and Beyond*, Addison-Wesley 1995, 592–620.

[19] N. Sakamoto, K. Ushijima. *Designing and Integrating Human Genome Databases with Object-Oriented Technology.* Proc. 5th DEXA 1994, Springer LNCS 856, 145–152.

[20] A. Spinelli, P. Salvaneschi, M. Cadei, M. Rocca. *MI — an object-oriented environment for integration of scientific applications.* ACM SIGPLAN Notices 29 (10) 1994, 212–222.

[21] E. Szeto, V.M. Markovitz. *ERDRAW 2.2. Reference Manual.* Technical Report LBL-PUB-3084, Information and Computing Sciences Division, Lawrence Berkeley Laboratory, CA, 1991.

[22] P. Taylor, V. Cahill, M. Mock. *Combining Object-Oriented Systems and Open Transaction Processing.* The Computer Journal 37, 1994, 487–498.

[23] P. van Voris, W. Millard, J. Thomas, D. Urban. *TERRA-Vision — the Integration of Scientific Analysis into the Decision Making Process.* International Journal of Geographical Information Systems, 7, 1993, 143–164.

[24] J.L. Wiener, Y.E. Ioannidis. *A Moose and a Fox Can Aid Scientists with Data Management Problems.* Technical Report 1182, Dept. of Computer Sciences, University of Wisconsin, Madison, WI, 1993.

[25] F. Zhan, D. Mark. *Object-Oriented Spatial Knowledge Representation and Processing: Formalization of Core Classes and their Relationships.* Proc. 5th International Symposium on Spatial Data Handling 1992, Vol. 2, 662–671.