Geo-Opera: Workflow Concepts for Spatial Processes

Gustavo Alonso Claus Hagen

Database Research Group
Institute of Information Systems
ETH Zentrum
CH-8092 Zürich, Switzerland
E-mail:{alonso,hagen}@inf.ethz.ch

Abstract. A Process Support System provides the tools and mechanisms necessary to define, implement and control processes, i.e., complex sequences of program invocations and data exchanges. Due to the generality of the notion of process and the high demand for the functionality they provide, process support systems are starting to be used in a variety of application areas, from business re-engineering to experiment management. In particular, recent results have shown the advantages of using such systems in scientific applications and the work reported in this paper is to be interpreted as one more step in that direction. The paper describes Geo-Opera, a process support system tailored to spatial modeling and GIS engineering. Geo-Opera facilitates the task of coordinating and managing the development and execution of large, computer-based geographic models. It provides a flexible environment for experiment management, incorporating many characteristics of workflow management systems as well as a simple but expressive process modeling language, exception handling, and data and metadata indexing and querying capabilities.

1 Introduction

The notion of *process* is currently being used in a variety of application areas to refer to complex sequences of computer programs and data exchanges controlled by a meta-program (the process). This idea has proven to be very helpful in solving a number of problems associated with environments involving heterogeneous platforms and applications. Some of these problems are: (1) interoperability, which is solved on a process basis instead of tackling the general case, (2) distribution, addressed by providing adequate support for programming distributed applications out of pre-existing tools, (3) forward-recovery, guaranteed by making every step of the process persistent using a database, (4) monitoring, based on the current and past states of the process stored in a database, and (5) history tracking, solved by providing a querying and data mining tool over the database storing the states of the processes. Surprisingly, these problems are pervasive and appear in many different application areas, from virtual enterprises and business environments [Fry94, LA94], to software engineering [CKO92, TKP94, BDMQ95] and scientific data management [ILGP96, BSR96].

Such pervasiveness explains the success of *workflow management* concepts and products, which are the most recent incarnation of process support systems.

Existing workflow tools, however, are often tailored to a particular domain. In particular, existing systems target in most cases either business processes [Hsu95] (this is the case of IBM FlowMark [MAGK95], for instance) or imaging systems (like OmniDesk [AAEM97]), with a few research prototypes addressing other areas [BK94, ILGP96]. Such narrow purpose design is combined with severe limitations related to performance and functionality [AS96, AAEM97], limitations that further restrict the applicability of these systems (see for example [MVW96, BSR96] reporting on attempts to use commercial workflow products to support scientific applications).

The project *OPERA* (Open Process Engine for Reliable Activities), currently ongoing at the Database Research Group of ETH Zürich, has as its main objective to paliate such glaring limitations of the state-of-the-art. OPERA is a workflow tool in the sense that it provides functionality that today only exists in workflow systems. OPERA is unlike existing workflow products in that it supports, from the same application platform, processes of very different nature. As an example of the potential offered by OPERA, this paper discusses in detail *Geo-Opera*, a tool, built as an extension of OPERA, supporting the development, execution and management of complex geo-models. Geo-Opera does not address requirements of spatial data handling such as indexing, storage, or efficient retrieval. For such purposes, Geo-Opera relies on existing systems. Geo-Opera is intended instead as a tool for the management of geo-processes, i.e., complex sequences of programs and transformations over spatial data. Geo-Opera supports all aspects of geo-process management, from the definition of geo-processes to their execution over heterogeneous hardware and software platforms, including as well querying and indexing capabilities over meta-data related to the geo-processes. An important contribution of Geo-Opera is bringing together under a single system functionality that was previously available only as part of isolated tools. It also introduces new concepts in geo-process management. For instance, the language used has a sophisticated exception and event handling mechanism. Since Geo-Opera also controls the execution of such geo-processes over distributed, heterogeneous environments, exception handling and events allow a greater flexibility when dealing with failures. Moreover, since a geo-process is a persistent entity in Geo-Opera, forward recovery and persistent execution is always ensured, which adds considerable guarantees when executing large and complex models involving many steps and, possibly, many computers. Finally, Geo-Opera provides a sophisticated mechanism for dependency tracking that can be used to derive the lineage of data sets, modifications to models, and perform automatic change propagation and notification. This mechanism also allows geo-processes to be modified creating new versions or to be executed repeatedly overwriting previous results or creating new versions of the resulting data.

With such functionality, Geo-Opera is a very powerful and flexible environment for the development, management, and execution of large geo-processes. This functionality is described in more detail in the sections that follow. Sec-

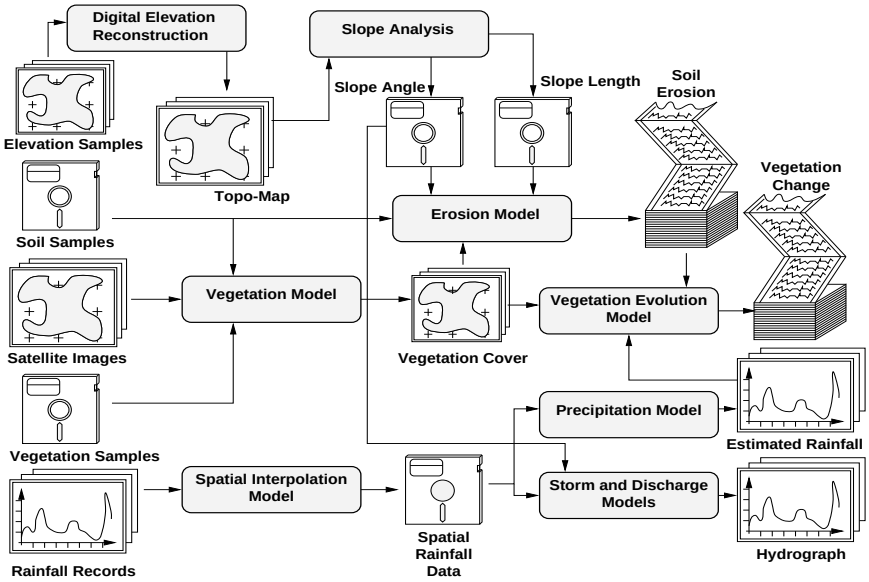

Fig. 1. Example of geo-process representing a geo-model of the hydrographic characteristics of a region

tion 2 presents an example of the applications to be supported by Geo-Opera as motivation for the rest of the paper. Section 3 discusses the architecture and functionality of OPERA. Section 4 describes how geographic process management can be performed using the facilities provided by Geo-Opera. Section 5 discusses related work and Section 6 concludes the paper.

2 Motivation and example

Most of the research activities in GIS and spatial data are centered around issues such as data representation, indexing, storage, and retrieval. To certain extent, this is the result of an attempt to extend database technology beyond traditional data sets (banking, insurance, financial) [vOV91, ZG91, Kea93]. These, however, are not the only problems related to spatial models [AE94a, AE94b, SSE+95], a point that is best illustrated with an example[1].

Figure 1 shows a combination of typical spatial models as sequences of transformations applied to different data sets (in what follows, the terms geo-model and geo-process will be used interchangeably. From an abstract point of view, a model is a representation of a geographic phenomenon while a geo-process is the

[1] This example is taken from [AE94b] where a solution to a related problem is proposed. Although the solution presented in this paper derives in part from that earlier work, the functionality and capabilities provided by Geo-Opera are more comprehensive.

model described in terms of programs and data sets). For instance, the elevation samples are used as input to a digital elevation reconstruction algorithm that produces a topographic map with interpolated elevations. This topographic map is then used as input to a slope analysis program that extracts the length, orientation and angle of the different slopes. These results, along with samples of the soil in the area and information about the vegetation cover, are used as the input for a model that tries to predict the future erosion patterns in the area. The vegetation cover is obtained by combining the information of the soil samples, vegetation samples and satellite images of the area. The vegetation cover is also used as one of the inputs of a vegetation evolution model that, given the predicted erosion, estimates the possible changes in vegetation cover. Similar step-wise procedures are followed to derive the required information regarding precipitation levels. The figure shows all these spatial models combined together into a complex set of operations representing different geographic phenomena.

This example illustrates several requirements that are very similar in nature to those related to process management. In what follows and for reasons of space, the paper will be focussed on a necessarily simplified list of requirements and how Geo-Opera provides functionality to meet those requirements. This list can be be summarized in three large areas: *modeling language*, *distribution and parallelism*, and *querying capabilities*.

Modeling Language . In any process support system, one of the key components is the language in which to express processes (geo-processes in this case). Given the intrinsic modularity of geo-processes (the example contains several geo-processes within a larger one), the language used must be structured, allow nesting, and facilitate reusability. Any geo-process should be suitable to be used as a building block for a larger geo-process. In addition, and on account of the complex execution environment, the language must provide support for event and exception handling. Events can be used, for instance, to notify the system of changes in the input data sets and trigger the execution of the model (or parts thereof) to produce more up-to-date results. Similarly, regarding exceptions, a reliable mechanism must be in place to cope with them to avoid that any deviation from the prescribed behavior results in having to abort the execution of the entire model. As in workflow systems, the language must allow defining and registering with the system external objects and applications since, especially in the case of geo-processes, both the algorithms and the spatial data will be external to the system. This registration of external entities is the basis for addressing interoperability issues (Point (1) above).

Distribution and Parallelism . To alleviate the cost of executing such complex geo-processes, their different steps should be executed in parallel whenever possible (Point (2) above). To avoid losing generality, we will assume the underlying platform is a cluster of PCs or workstations, not necessarily using the same operating system. Each step of the geo-process can be assigned to a different node in the cluster, thereby exploiting the parallelism inherent to the model. The same ideas can be applied in a multiprocessor machine (in which homogene-

ity greatly simplifies the problem). As a direct consequence, and due to the cost of the models and the time it takes to complete their execution, a mechanism must be in place to avoid losing all computations when a failure occurs. This is the notion of forward recovery (Point (3) above), in which execution can be resumed from the point where it was interrupted by a failure. Moreover, it does not help to provide a system in which complex geo-process are executed from beginning to end without being able to stop execution at intermediate points, or to dynamically modify the geo-process to correct errors. Often, a geo-process is built not so much to test its results but to test the validity of the corresponding geo-model. For these cases, the system should support step-by-step execution and provide the ability to, at any point, stop, examine the geo-process, make changes, and resume execution. Such functionality can only be provided through close monitoring of the geo-process execution (Point (4) above).

Querying Capabilities . Geo-processes result in derived data that cannot be interpreted without knowledge about the model used for its creation and the initial data used in the model [Arm88, Lan88, LV90]. This leads to the well known problems of lineage-tracking, change propagation and versioning [Tob79, GG89, Rad91, SSAE93] which are basically no different from those of tracking the history of more general processes (Point (5) above). Questions such as "which models use algorithm X", "which results may change if dataset Y is updated", and "which data sets are used to derive result Z" are typical of such environments. Moreover, operations such as automatic change propagation (re-execution of a model when the input data is modified producing new versions of the output data), change notification, and process control (maintaining a copy of the model used to create a data set for as long as the data set exists, and maintaining copies of subprocesses for as long as there are other processes using them) must be supported by the system to make it truly useful. This functionality can be provided only if there are adequate mechanisms to track dependencies and efficient ways of retrieving information about these dependencies.

3 OPERA: a Process Support System

The OPERA project is an example of the ongoing efforts towards generalizing workflow concepts. OPERA is an open and flexible process support system intended to work as an specialized distributed operating system for large, heterogeneous, distributed environments, in which OPERA takes over the tasks of scheduling, synchronization and load balancing complex applications.

3.1 System Architecture

The architecture of OPERA is intrinsically modular. Parts of it have been implemented using TP-monitors and other middleware solutions. The current prototype relies on Oracle and ObjectStore as the underlying databases, and on Encina as the transaction engine. Future versions may incorporate functionality

from CORBA related products, and queuing systems. For reasons of space, only the aspects of OPERA relevant to this paper are discussed.

The system architecture of OPERA consists of three service layers, shown in Figure 2: *database services*, *process services* and *interface services*. The idea is that each service layer should be independent and easily replaceable, an important point since OPERA is a generic process support system that needs to be tailored to specific applications. The components within each layer are also built as separate modules so they can be discarded or included in the final system depending on the requirements.

The database service layer provides persistent storage. It is divided into five *spaces*: template, instance, object, history, and configuration spaces, each one of them storing a different type of process data. This approach enhances scalability as it allows to install each space on a different physical database and on a different machine to prevent, for instance, the traffic generated during process execution from interfering with the traffic generated by history analysis. The logical organization of these spaces corresponds to the main entities handled in OPERA.

Templates are similar to the process code image. The template space is used to store the process defined and to create new processes, effectively separating development of processes from process execution. Instances are running processes. For each running instance of the same process, the instance space contains a copy of the process image that is updated as the process executes. Storing this information persistently guarantees forward recoverability in case of failures, since the system can resume operations as soon as failures are repaired. If more availability is needed, there are several techniques that can be used [KAGM96].

Objects are used to store information about externally defined data. This space forms the basis to allow OPERA to interact with other applications and it also stores part of the information related to the dependencies among objects. The history space is used to store information about already executed instances, which allows to query the system for any result of past executions. Finally, the configuration space is used to record system related information such as configuration, internet addresses, program locations, and so forth.

An important additional aspect of OPERA is that it is database independent thanks to the *DB abstraction layer* which translates a canonical representation [KAGM96] to the private representations of the underlying repositories. OPERA handles the contents of the five spaces using its own internal representation (see below), which has been optimized for performance and expressibility. The *DB abstraction layer* translates this internal representation to the appropriate language (SQL, C++, system calls) as required by the physical implementation of the underlying database.

The process service layer provides the tools necessary for coordinating and monitoring the execution of processes, as well as the tools for defining and creating new processes. The most relevant components are the *dispatcher*, the *navigator*, the *object manager*, and the *query manager*.

The dispatcher plays the role of resource allocator, determining which system

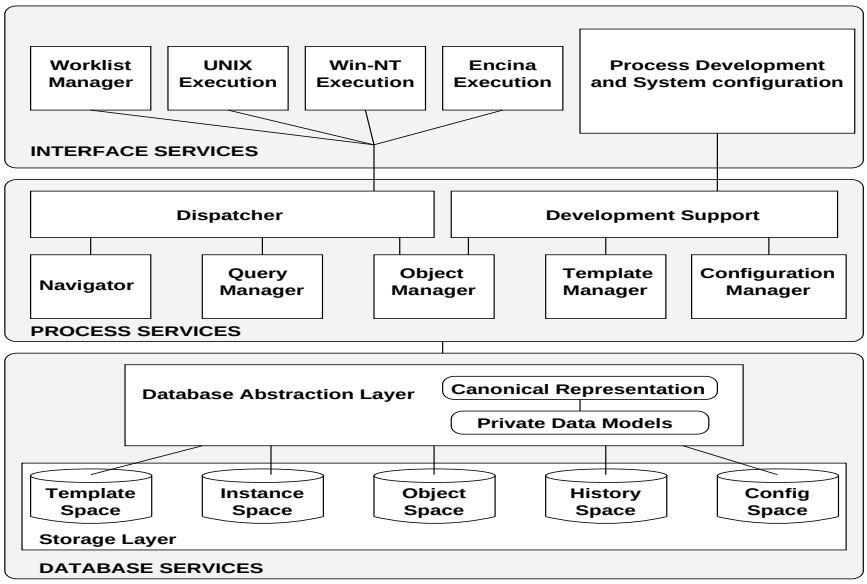

Fig. 2. System architecture of OPERA

will execute the next step, locating a machine with the appropriate characteristics, checking whether it is currently available, and managing the communications with remote system components. The navigator acts as a scheduler: it "navigates" through the process description establishing which steps should be executed next, which ones are to be delayed, and which ones have finished their execution. During this navigation procedure, the related variables in the process instance are updated so as to have a faithful record of every step taken in the execution of the process.

The object manager is used for maintaining up to date the information related to external objects. As it will be explained in more detail below, this includes keeping track of which processes use a particular object, the lineage of the object if it is a derived one, and avoiding inconsistencies in the version being used. The query manager has as its main role to provide a suitable interface for querying information about past and current processes. The query manager reroutes queries to the space involved and merges results to present a coherent view to the requesting entity. The query manager is actively used not only by the final user but also by the system itself to find out existing dependencies between processes and objects.

The interface service layer provides the mechanisms necessary to interact with users (through the *worklist managers*), and applications (through the *execution interfaces* supporting several operating systems and specific tools). The execution interfaces are divided into two parts, one internal to OPERA and used for establishing the correct interaction protocol with external applications, and one external to OPERA and residing in the same machine where the application

being invoked resides. In this, OPERA follows closely the architecture of existing workflow management systems. In addition, although it will not be discussed further in this paper, a number of tools are provided for defining new processes as well as for monitoring the system's configuration.

3.2 OPERA Execution Environment

The main goal of OPERA is to provide an environment in which complex processes can be executed. Thus, the main hardware platform targeted by OPERA is a cluster of workstations or PCs, linked by a LAN or WAN. Such a platform is used as a shared nothing multi-processor environment in which OPERA plays the roles of scheduler and resource allocator. OPERA relies on a database for storing the information it needs to perform its tasks but, unlike existing workflow systems, OPERA does not rely on a unique, centralized database. OPERA takes advantage of the underlying hardware platform to distribute its functionality. Thus, all OPERA components can be moved from node to node in the system, including the database with the system information. This makes OPERA very robust and helps to address difficult issues such as reliability, availability, and scalability. OPERA is capable of relying on a variety of databases both relational and object oriented, and it can invoke applications under both UNIX and NT. It can also invoke applications developed in specific systems such as TP-Monitors (Encina, for example) and provides the necessary hooks to create additional execution interfaces for other environments, thereby maintaining an open architecture. In this, OPERA is no different from existing workflow management system. OPERA, however, has generalized many of the existing solutions and paid special attention to avoiding the pitfalls and design limitations found in current workflow products [AS96, AAEM97].

4 Geo-Opera: a Geo-Process Support System

Geo-Opera is a customization of OPERA for geo-processes, or using recent terminology, for experiment management in geographic applications [ILGP96]. The particular characteristics of geo-processes make it necessary to extend OPERA with functionality not strictly related to generic processes. This section discusses such functionality and also illustrates how Geo-Opera can be used to address the issues discussed in Section 2.

4.1 From User Representations to Data Models

Geo-Opera takes advantage of the open architecture of OPERA to provide a modeling language suitable for geographic applications. OPERA contains a hierarchy of process representations rather than a single model. Figure 3 describes this hierarchy of languages and representations, along with its current status in Geo-Opera. At the top, and used at the interface service layer, is the application specific language. This is the language that has been customized for Geo-Opera.

As in most process support systems, this language has a strong graphic component and, from the user's point of view, a geo-process is represented through icons not very different from those shown in Figure 1. In addition to this mostly graphical language, there are a number of interfaces to allow the user to specify any additional information necessary to execute a geo-process. This ranges from registering external objects to indicating in which machines a particular algorithm can be invoked.

User representations, however, are not suitable for handling processes efficiently. As an intermediate representation, used at the process service layer to facilitate interoperability across heterogeneous platforms, there is the *Opera Canonical Representation* (OCR). OCR is not so much a language but a data model or internal representation that provides a unique way to identify all entities used in the process support system. It provides features such as rules, event handling, exception handling, sophisticated programming constructs, and complex representations,, not all of which are necessarily used in a given application. For instance, many of the programming constructs built in OCR are not used in Geo-Opera (for instance, business processes often need to describe complex and open-ended negotiations between tasks that do not occur in geographic models). Finally, since the different storage spaces correspond to actual databases with their own private data models, OCR must be translated into these data models (ObjectStore and Oracle [KAGM96]).

4.2 Expressing Geo-processes in Geo-Opera

In its first version, the Geo-Opera language is a simplification of the languages traditionally found in commercial workflow management systems (see [KAGM96] for an example). The simplification is possible due to the intrinsic nature of geographic modeling in which most of the modeling logic is in the individual algorithms. Thus, constructs such as start and end conditions, as well as control flow conditions are not strictly necessary although, since they are supported by OCR, they could be easily incorporated if needed. The main components of the Geo-Opera language are:

Models (also called *projects* in GOOSE [AE94b], or *processes* in most workflow systems). A model corresponds to a geo-process and consists of a collection of *tasks* linked by *connectors* along with the so-called *blackboard*. The tasks are either activities (basic units of execution), blocks (groupings of tasks with special properties), or submodels (references to other model descriptions, late binding is used to instantiate submodels of a running model). The connectors specify order precedences between the different tasks. The blackboard is used to store intermediate results and exchange data between tasks. It fulfills the same role as the control data in workflow systems, the only difference being that all control variables are available in a single place, the blackboard, while in workflow systems this data is often found in the private data containers of each task.

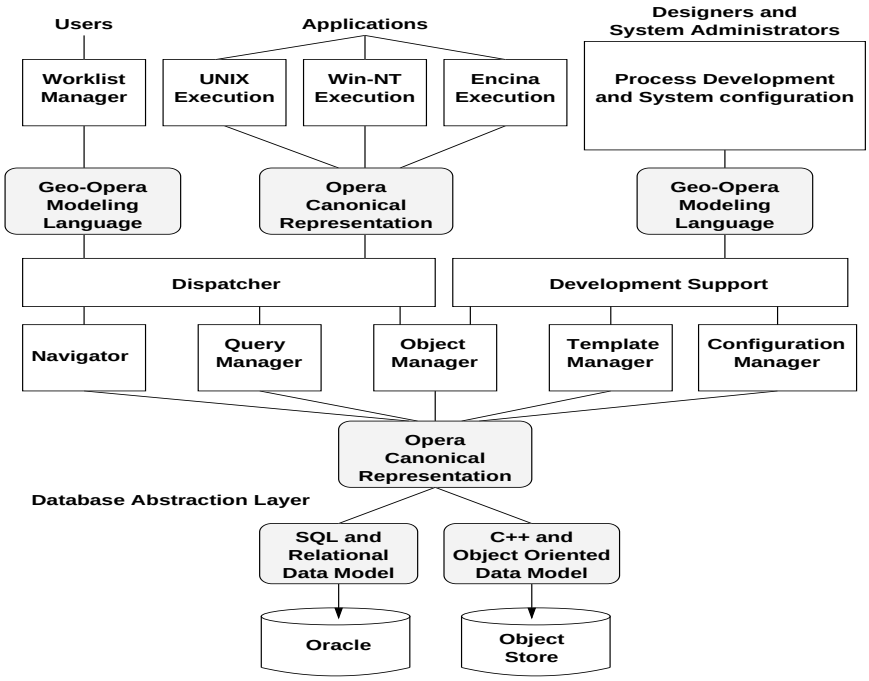

Fig. 3. Language and Representation Hierarchy in Geo-Opera

Tasks (also called *activities* or *steps* in some workflow systems). Tasks are internally represented in Geo-Opera as objects. This provides a well-defined interface to allow starting and stopping execution, to provide call parameters, to query execution state and to return values. Each task has a well defined set of attributes and methods in order to facilitate basic navigation. This set is defined through the *task interface* and inheritance is used to allow hierarchical structuring of task interfaces. This allows to extend the system by adding new task interfaces. At the root of the tasks inheritance tree there is a special interface called BasicTask, which provides a minimal set of functionality. The structure of the basic task interface is given in Table 1. The different task types (submodels, blocks, and activities for our purposes here) allow nesting and modular design of models. Submodels correspond to other models which have been independently defined (similar to a library call). Blocks correspond to logical constructs within a model, they have no name and can only be used once (similar to bracketing a set of instructions between *BEGIN* and *END*). Activities are the actual programs to be invoked. These programs are external to the system and must be registered before they can be used (as it is done in any workflow system).

Control Flow . In the Geo-Opera language, control flow is specified through simple connectors between tasks. Thus, the user only needs to link two tasks with a connector to establish the order of execution. Internally, this control connectors

Name	Type	Description
Guard	Guard	Control flow description
InBox	Parameterlist	Call parameters
OutBox	Parameterlist	Allows to access return values
State	ExecutionState	Allows to access the current state of task execution
ExceptionDecl	List of Exception	Declares the exceptions that can possibly be thrown by the task
EventQueue	List of Event	Allows to query the list of events raised by the task
Type	TaskType	Distinguishes between submodels, blocks, and activities
Reference	Object	A reference determining how the task is to be executed

Table 1. Basic Task Interface

do not exist, control flow inside a model is specified only through the guards of the corresponding tasks. A *guard* describes when the task has to be executed. The paradigm used is similar to that known as *ECA rules* in active databases [WC96]. The guard consists of an *activator* that specifies when the task has to be considered for execution, and a *start condition* (the latter is not used in the case of Geo-Opera, but is part of OCR). The activator is a predicate that can reference the execution state of other tasks, events raised by other tasks, and imported events of the process. Note that only tasks within the same process are visible to the guard. This principle of locality guarantees efficient evaluation of guards and is an important difference to active database systems, where all events are visible to all rules. Figure 4 shows how the example of Figure 1 looks like using tasks and control connectors.

External Objects . Data items not residing within Geo-Opera must be registered before they can be used (in the same way that in a workflow management system, applications and users must be registered with the system before they can be involved in process execution). The registration process involves specifying all the necessary parameters so as to allow the system to access and extract the corresponding data. This includes the user provided filter (see below for more details). Additional attributes of the external object allow the system to keep track of the dependencies established with other objects and models. These attributes are maintained by the system and are not directly accessible to the user. The list of attributes of a basic external object is shown in Table 2.

Exception Handling . Exception handling is necessary to be able to cope with all the possible errors that may occur during the execution of a model. It is not reasonable to expect that the programmer of a model will foresee all possible

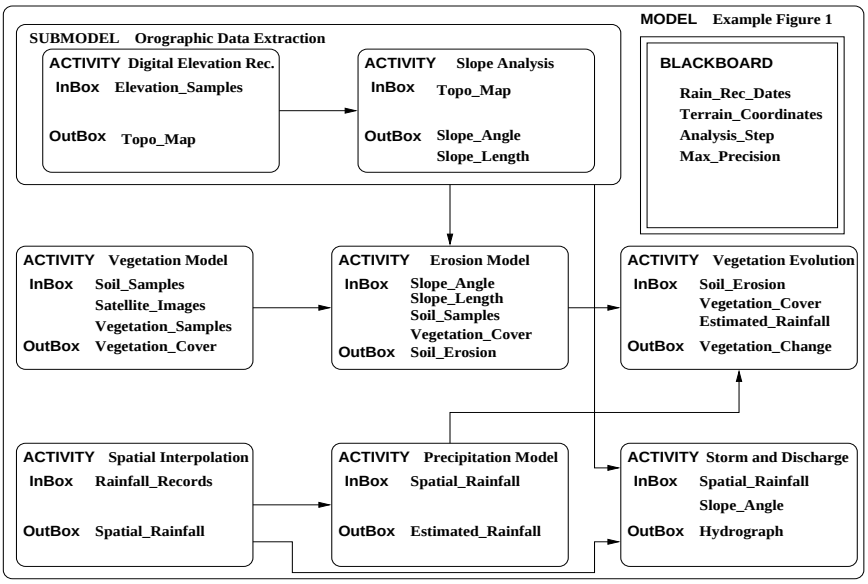

Fig. 4. The Example of Figure 1 in terms of activities, submodels and control connectors.

exceptions and hence, a mechanism must be in place to cope with such cases. OPERA incorporates an extension to workflow languages to support the flexible, transparent modeling of fault-tolerant workflows and has a strong focus on reusability. This extension is based on exception handling concepts from programming languages and uses a default hierarchy of exception handlers to allow an exception to always be trapped, if by nobody else, at least by the system. This approach has the advantage of always providing an opportunity to correct the exception and resume execution before having to abort the execution.

Events . Events are used both to interact with external entities and to allow interactions between models executed concurrently. As mentioned above, events can be associated with the guards of a task. In this way, a modification to an external object triggers an event that can be captured by the system which will proceed accordingly (see below the discussion on active mechanisms). Events also allow external applications to control the process support system: by generating different events, an application can control the execution of a model. For reasons of space, these two issues (exception handling and events) will not be discussed further.

4.3 Distribution and Parallelism

The execution of geo-processes requires expensive computations. Often, the data does not fit in main memory (as it is the case with many raster images) which

Name	Type	Description
ObjectName	String	User given name
OID	Long Integer	System generated object identifier
System	String	System under which the data item is located (UNIX, Win-NT, etc.)
Location	String	System address to use when locating the object
Access	String	Access mask to use to retrieve the data (the user provided filter)
Sources	List of OID	Other objects used to create this object, only the immediate ancestors are stored (search is recursive)
Versions	List of OID	Other versions of this object
Algorithm	Task	The task used to create this object
Usage	List of Task	Tasks that use this object as input (OID is in Task.InBox)

Table 2. External Object Interface

implies expensive I/O, and the operations tend to be complex transformations. Geo-Opera allows to alleviate this problem by supporting the execution in parallel of as many activities as the data flow dependencies in the model allow. In this, Geo-Opera follows the trend towards using clusters of workstations as the basic hardware infrastructure. The intrinsic distributed nature of processes, exploited in business applications to increase the decentralization of the organization, is used in Geo-Opera to separate the execution of every single task. As long as the hardware resources and necessary communications are available, each activity of a model can be executed in a different machine. This idea is shown in Figure 5 in which the model of Figure 1 is executed in a cluster of 5 workstations using a shared file system. The execution of the model requires only three steps, as activities that can be executed concurrently are assigned to different workstations and run in parallel. The use of a common file system (such as NFS, found in most UNIX environments) simplifies a great deal the problem of data handling. It is not necessary, however, to have such shared file system. Geo-Opera can operate and execute activities in parallel also over wide area networks (as most workflow management systems do) but this implies that data will have to be accessed remotely which can have a considerable impact on performance (see [ARM97] on how to cope with data handling problems in these environments).

4.4 External Objects

Handling external data is crucial in geo-processes. Most geographic and spatial data resides in files and not in databases. This creates an additional problem for there is no obvious way to interact with data outside the system. To address

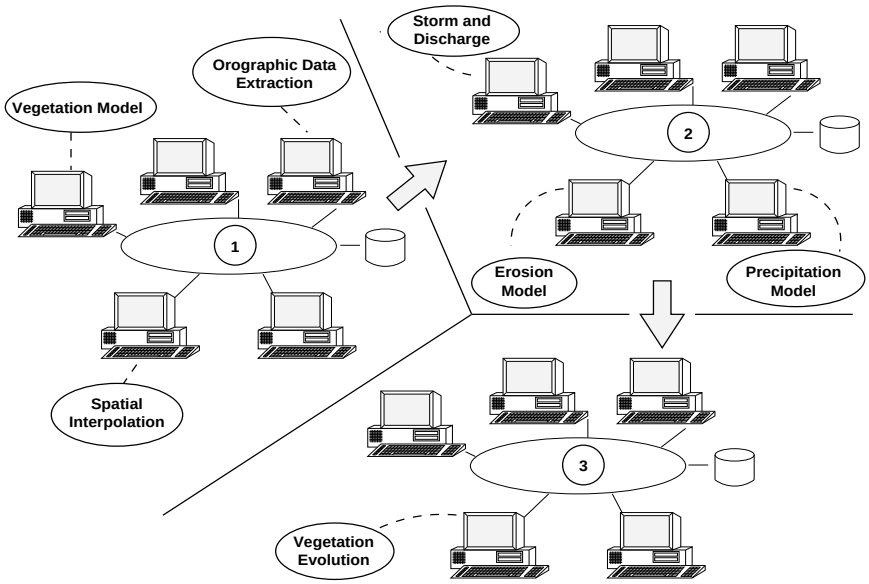

Fig. 5. The model of Figure 1 executed in a distributed fashion in a cluster of workstations; in each phase, several tasks may be executed in parallel.

this interesting issue, Geo-Opera makes use of the ideas described in the GOOSE system [AE94b] and recent contributions from the area of externalized database functionality, in particular from the Concert prototype [BRS96]. The basic idea is to treat external objects as black boxes over which database functionality can be applied. From the Concert prototype, Geo-Opera uses the ability to export storage management services such as indexing, replication, and basic query processing over externally defined data. From the GOOSE system, Geo-Opera has borrowed the object representation mechanism as well as most of the object handling capabilities (*object base* in GOOSE). Accordingly, the *external objects* supported by Geo-Opera are data objects located in external repositories such as file systems, databases, or the internal storage of a particular applications. An external object is a *view* over external data, i.e., it is possible to have several Geo-Opera external objects that refer to the same actual data item. A standard example is that of one external object defined over the chemical properties of a set of soil samples, and a second object defined over the physical properties of that same set of soil samples. Such views are computed through a user-provided *filter* that is activated when the object is accessed.

External objects must be registered in the instance space. A user wishing to define an external object has to provide a unique name (under which it is referenced in process descriptions) and the filter that describes how the object is constructed from the external data source. A system address is also necessary to indicate where Geo-Opera can find the data set (this can be either a network address or a symbolic name such as those used in environments like CORBA or

DCE). The user provided filter can be a database query or a sequence of operating system commands extracting data from a series of files. This is irrelevant from the point of view of Geo-Opera since the only role of the system is to invoke the filter to get the data. Note that this filter is no different from an algorithm or a task as defined in geo-processes. In fact, Geo-Opera uses the same mechanisms to invoke the filter of an external object and the program corresponding to a geo-process' activity. Also note that, through this filters, Geo-Opera can incorporate external repositories without having to address the general interoperability problem. If there is a uniform way to extract information from an external repository (like an SQL interface, for instance), this can be used for all accesses to the repository. For legacy systems, the filter option provides a quick and *ad-hoc* solution for those cases in which there is no need to implement a uniform access mechanism.

To allow the system to keep track of modifications produced by the execution of geo-processes, external objects can be versioned. This is accomplished by assigning a new version number to the objects resulting from executing a task and maintaining the corresponding attributes of all related objects up-to-date (see the "Versions" attribute of an object in Table 2). In the same way, Geo-Opera keeps track automatically of an object's *lineage*, recording by which processes it has been produced and which inputs have been used for its computation (see the "Algorithm", "Usage", and "Sources" attributes in Table 2). This information is heavily used by the active mechanisms and by the query manager.

4.5 Active Mechanisms

One of the most complex tasks of an experiment management system is to automate the tedious operations of versioning and updating. In most cases this is done manually, which adds considerable overhead and is an error prone process. Geo-Opera provides the notion of *active objects* and *active models* to address this issue. Active objects are automatically recomputed if some object they depend on has been modified. To avoid expensive checks, the update takes place in a lazy manner. Instead of doing a search of all related objects every time an object is modified, the update takes place when an object is accessed. The underlying mechanism is based on the attributes added to both objects and models. When an object is modified, a flag is set in all related objects. This takes only a few operations and does not interfere with the actual computation. When an active object is accessed, the flag is checked. If the flag is set, it means there has been a modification of the sources used to create the object, i.e., the current version is not the most up-to-date version. In such cases, the lineage chain is retraced and executed again to produce a new version. Non-active objects use this flag to notify the user of possible inconsistencies. This mechanism is similar to *backward chaining* in Marvel [TKP94]). Figure 6 shows the lineage of an object (from the model shown in Figure 1) and how this lineage can be used for the active object mechanism. Note the double linked list effect of the attributes, a characteristic exploited for more effective querying and mining.

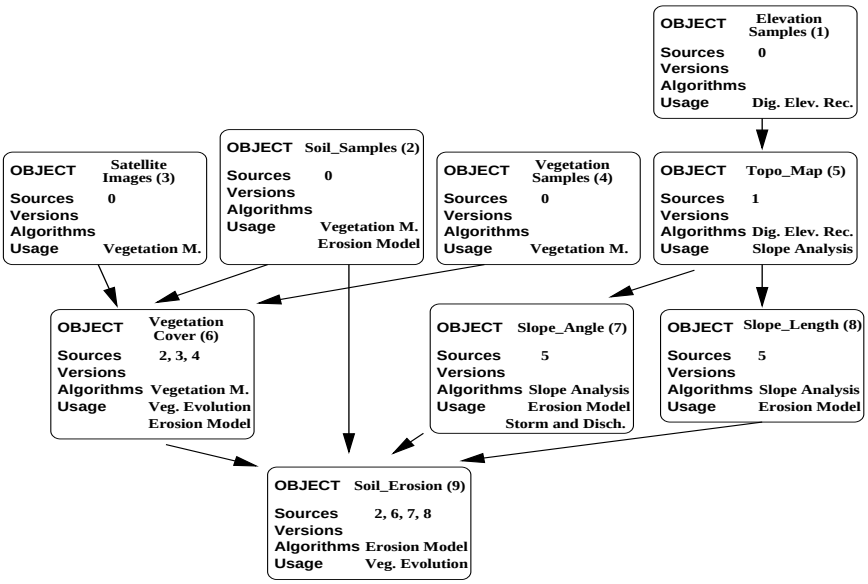

Fig. 6. Active objects and automatic change propagation.

Similarly, active models are those that must be recomputed when some of their inputs are modified. This allows to get derived data automatically every time new input is available. As with active objects, the mechanism is recursive but, unlike active objects, is based on eager propagation, i.e., the model is recomputed every time one of the inputs changes. If an active model produces new versions of objects that are input to other active models, these projects are triggered as well (similar to *forward chaining* in Marvel [TKP94].) The active object and active project facilities are complementary and give the process designer fine-grain control over the system's behavior. The active object mechanism provides a form of lazy evaluation and is thus well-suited for objects that are seldom read but whose inputs change very often. The active project mechanism, on the other hand, supports fast updates to derived objects. Combined with the event handling mechanisms, this is a very powerful tool to perform complex computations over a stream of data provided by external sensors.

Geo-Opera also adds a versioning mechanism *for programs*. Versioning has been used in a variety of systems for manipulating data. In Geo-Opera, each individual task within a process corresponds to a program. In business applications these programs do not change often and, if they do, the older versions can be immediately discarded. In scientific applications, modeling is done not only to obtain data but also to test the model itself, i.e., whether the perception of natural processes reflected in the model corresponds to reality. This implies running many versions of the same geo-process adjusting individual tasks until the results are satisfactory. For the same reasons that the dependencies between a process and its related data must be kept, the different versions of a task and

a geo-process must be preserved so as to allow the user to repeat experiments, readjust parameters, and iterate over several design phases. In this regard, Geo-Opera is no different from a CASE environment in which versioning of code is a common feature. The way in which this versioning takes place is very similar to the procedures followed for external objects (in fact, some information about external programs is stored in the same form as the external objects).

4.6 Querying Capabilities

An important feature of Geo-Opera is its query management facilities. Aside of the standard process query capabilities common to all process support systems, Geo-Opera also tracks dependencies and versions of both data (internal and external) and processes. Most of these mechanisms have been outlined above and are based on the information stored about the external objects and the geo-process instances. Unlike in business applications where the process itself is usually of no importance once the final result is achieved, in many scientific applications, including the ones targeted by Geo-Opera, data cannot always be understood without the process that was used to create it. Examples of this problem in genetic experiments are well known [BSR96]. There are very strong ties between a process and the data resulting from the process, ties that must be maintained for as long as the data or the process remains in the system. In Geo-Opera, most operations result in a number of checks being performed to avoid deleting data or processes related to other data or other processes. This capability is not present in neither workflow management systems or GISs. The querying capabilities are extensively used to find information about object lineage (what objects are ancestors of another object), process and object dependencies (which processes use which objects, which objects are used in a process), and, in general, by any of the active mechanisms. Needless to say that these facilities are also made available to the end user, but it is important to point out that they are also used by the system itself to avoid inconsistencies.

As an example of the role played by the querying facilities, consider the problem of calculating the error associated to a particular data set. This error is derived by some complex manipulations and combinations of the errors associated with all the ancestors of the object. An algorithm trying to establish this error, needs to be able to access the lineage of the object. Thus, the user first retrieves the lineage of the object (which is calculated by the system by following the chain of "Source" attributes), then it retrieves the algorithms used in each step (following the algorithm attribute of each object). With this information, it can then try to establish the resulting error from the overall computation. A second example, this time on how Geo-Opera uses this query facilities, is the active mechanism. If a geo-process modifies an object, this is recorded in the system by creating a new version. When an object is notified that a new version exists, the "Usage" attribute is used to determine which tasks have this object as an input. Accessing the information about these tasks, Geo-Opera can determine whether it is necessary to execute them again. It can also set a flag in all objects derived from the original object indicating that a new version exists. The lists

of derived objects is obtained by recursively following the "Usage" attribute of the object and the "OutBox" attribute of the tasks. In both examples, the interaction with the corresponding spaces where this information is stored (instance, history, and configuration spaces) is handled by the query manager. To increase performance, the query manager provides a number of entry points (for operations such as lineage, process recalculation, and inter-process dependencies) that are used to trigger all the queries necessary to solve the request in question.

5 Related Work

Geo-Opera is motivated by the similarity between geo-processes and generic processes. Some of the common issues were already hinted at by early work in the area [SSE+95, AE94a, AE94b, SW93, HGM93, HQGW93] but it has been only recently that workflow systems have been suggested as an adequate tool for scientific data management [MVW96, BSR96]. In addition, some work has been done in the experiment management area [ILGP96] with close ties to workflow notions. All of this research has provided insights on how process support systems may be used in scientific applications. In particular, ZOO [ILGP96] is a *desktop experiment management environment* for different scientific disciplines. It consists of a generic tool that can be extended by customized enhancements to make it suitable for different application domains. ZOO is based on a scientific database for experiment data and provides services for modeling, automation of experiment execution, and analysis of results. Conceptually and from an application point of view, Geo-Opera would be similar to an extension of ZOO for geographic modeling. In practice, however, there are significant differences between both approaches. OPERA is a generic process support system while ZOO is a dedicated system supporting scientific experiments. ZOO does not provide a generic notion of process and the tailoring refers to adapting the system to a particular scientific discipline, not to a given type of processes. Since much of the functionality provided in Geo-Opera applies to other scientific disciplines, it is conceivable to develop modifications of Geo-Opera for other type of scientific modeling. Moreover, ZOO models processes implicitly through dependencies between objects in an object-oriented data model. There is no explicit modeling of control flow and a centralized database is used for storage. Thus, ZOO lacks the extendibility and modularity of OPERA, which can employ arbitrary databases (even simultaneously) for storage and to manage external objects through the view mechanism.

The WASA project [MVW96, MVW95] is an attempt to use a commercial workflow management system for supporting scientific work. WASA allows the modeling, execution, and analysis of experiments in different scientific disciplines. Like GEO-Opera, WASA supports access to data stored in repositories external to the system (through so-called object brokers). There is, however, no support for the modeling of dependencies between objects and the automatic recomputation of models. The Geo-Opera architecture differs from the WASA approach further in its focus on distribution, heterogeneity, and scalability - issues which

we believe are of crucial importance in these applications.

Geo-Opera has borrowed a number of solutions from GOOSE [AE94b] and CMS [SSE+95]. GOOSE is a tool for scientific experiment management, less developed than ZOO from an architectural point of view, but with a more powerful modeling mechanism in terms of dependency tracking. The notion of external objects and the way they are handled in Geo-Opera follows closely the ideas described in GOOSE. GOOSE, however, was never intended as a generic tool and it would be very difficult to adapt it to applications beyond those for which it was originally intended

Finally, Process Centered Environments, PCE, support process modeling and execution in software engineering applications [BK94, DG90, TKP94]. While these systems provide some of the data handling facilities present in Geo-Opera (namely the modeling of object dependencies and the automatic recomputation), the generality of the view mechanism is missing. In addition and due to the very different nature of the application domain, Geo-Opera provides functionality that cannot be expected from a software engineering tool (although it might be possible to extend these tools to provide functionality resembling that of Geo-Opera).

6 Conclusions

Geo-Opera is an extension of a generic process support system, OPERA, tailored to geographic and spatial modeling. Geo-Opera takes advantage of workflow technology to provide a fully distributed, heterogeneous computing platform in which to develop, execute, and manage complex geographic models. Geo-Opera, however, goes well beyond workflow management systems by providing ad-hoc functionality for tracking data dependencies and adapting them to the particular needs of scientific modeling. We see Geo-Opera as a first step towards bridging the gap between the potential offered by clusters of workstations and real applications. In many ways, Geo-Opera can be seen as an specialized distributed operating system for scientific applications in general and for geographic modeling in particular. Geo-Opera is currently in the prototyping stage, although much of the functionality described in the paper has already being tested and implemented. Future work includes enhancing the functionality of Geo-Opera by building explicit links to commercial GIS. We also plan to explore extensions of both OPERA and Geo-Opera to other application areas.

References

[AAEM97] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and Limitations of Current Workflow Management Systems. *IEEE Expert*, 12(5), September-October 1997.

[AE94a] G. Alonso and A. El Abbadi. Cooperative Modeling in Applied Geographic Research. In *Proceedings of the International Conference on Cooperative Information Systems, CoopIS'94, Toronto, Canada*, May 1994.

[AE94b] G. Alonso and A. El Abbadi. Cooperative Modeling in Applied Geographic Research. *International Journal of Intelligent and Cooperative Information Systems*, 3(1), May 1994.

[Arm88] M.P. Armstrong. Temporality in spatial databases. In *Proceedings GIS/LIS*, pages 880–889, November 1988.

[ARM97] G. Alonso, B. Reinwald, and C. Mohan. Distributed Data Management in Workflow Environments. In *Seventh International Workshop on Research Issues in Data Engineering (RIDE'97)*, Birmingham, England, April 1997.

[AS96] G. Alonso and H.-J. Schek. Database Technology in Workflow Environments. *INFORMATIK/INFORMATIQUE (Journal of the Swiss Computer Science Society)*, April 1996.

[BDMQ95] A. Bernstein, C. Dellarocas, T.W. Malone, and J. Quimby. Software Tools for a Process Handbook. IEEE Computer Society, March 1995.

[BK94] I.Z. Ben-Shaul and G.E. Kaiser. A paradigm for decentralized process modeling and its realization in the oz environment. In *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, 1994.

[BRS96] Stephen Blott, Lukas Relly, and Hans-Jörg Schek. An open abstract-object storage system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, June 1996.

[BSR96] A. Bonner, A. Shrufi, and S. Rozen. LabFlow-1: A Database Benchmark for High Throughput Workflow Management. In *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT96), Avignon, France*, March 1996.

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

[DG90] W. Deiters and V. Gruhn. Managing software processes in the environment MELMAC. In *4th ACM SIGSOFT Symposium on Software Development Environments*, 1990.

[Fry94] C. Frye. Move to Workflow Provokes Business Process Scrutiny. *Software Magazine*, pages 77–89, April 1994.

[GG89] M. Goodchild and S. Gopal. *Accuracy of Spatial Databases.* Taylor and Francis Ltd, 1989.

[HGM93] N.I. Hachem, M.A. Gennert, and Ward M.O. The Gaea System: A Spatio-Temporal Database System for Global Change Studies. In *AAAS Workshop on Advances in Data Management for the Scientist and Engineer, Boston, Massachussetts*, pages 84–89, February 1993.

[HQGW93] N.I. Hachem, K. Qiu, M. Gennert, and M. Ward. Managing Derived Data in the Gaea Scientific DBMS. In *Proceedings of the 19th International Conference on Very Large Databases, Dublin, Ireland*, 1993.

[Hsu95] M. Hsu. Special Issue on Workflow Systems. *Bulletin of the Technical Committee on Data Engineering, IEEE*, 18(1), March 1995.

[ILGP96] Y.E. Ioannidis, M. Livny, S. Gupta, and N. Ponnekanti. ZOO: A desktop Experiment Management Environment. In *Proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India*, September 1996.

[KAGM96] M. Kamath, G. Alonso, R. Günthör, and C. Mohan. Providing High Availability in Very Large Workflow Management Systems. In *In Proceedings of the Fifth International Conference on Extending Database Technology (EDBT'96)* , Avignon, France, March 1996. Also available as IBM Research Report RJ9967, IBM Almaden Research Center, July 1995.

[Kea93] R.H. Katz and et al. Design of a Large Object Server Supporting Earth System Science Researchers. In *AAAS Workshop on Adavances in data Management for the Scientist and Engineer, Boston, Massachussetts, USA*, pages 77–83, February 1993.

[LA94] F. Leymann and W. Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal*, 33(2):326–348, 1994.

[Lan88] G. Langram. Temporal GIS design tradeoffs. In *Proceedings GIS/LIS*, pages 890–899, November 1988.

[LV90] D.P. Lanter and H. Veregin. A Lineage Meta-Database program for propagating error in Geographic Information Systems. In *Proceedings GIS/LIS*, pages 144–153, November 1990.

[MAGK95] C. Mohan, G. Alonso, R. Günthör, and M. Kamath. Exotica: A research perspective on workflow management systems. *Bulletin of the Technical Committee on Data Engineering, IEEE*, 19(1), March 1995.

[MVW95] C.B. Medeiros, G. Vossen, and M. Weske. WASA: A workflow-based architecture to support scientific database applications. In *Proc. 6th DEXA Conference*, London, 1995.

[MVW96] J. Meidanis, G. Vossen, and M. Weske. Using Workflow Management in DNA Sequencing. In *Proceedings of the 1st International Conference on Cooperative Information Systems (CoopIS96), Brussles, Belgium*, June 1996.

[Rad91] F.J. Radermacher. The Importance of Metaknowledge for Environmental Information Systems. In *Proceedings of the 2nd Symposium on the Design and Implementation of Large Spatial Databases, Springer Verlag*, volume 1, pages 35–44, August 1991.

[SSAE93] T.R. Smith, J. Su, D. Agrawal, and A. El Abbadi. Database and Modeling Systems for the Earth Sciences. *IEEE Bulletin of the Technical Committee on Data Engineering*, 16(1):33–37, March 1993.

[SSE+95] T. Smith, J. Su, A. El Abbadi, D. Agrawal, G. Alonso, and A. Saran. Computational Modeling Systems. *Information Systems*, 20(2), 1995.

[SW93] H.J. Schek and A. Wolf. From Extensible Databases to Interoperability between Multiple Databases and GIS Applications. In *Proceedins of teh 3rd Int. Symposium on Large Spatial Databases*, Singapore, June 1993.

[TKP94] A.Z. Tong, G.E. Kaiser, and S.S. Popovich. A flexible rule-chaining engine for process-based software engineering. In *Proceedings of the Ninth Knowledge-Based Software Engineering Conference*, Monterey, CA, USA, 1994.

[Tob79] W.R. Tobler. A transformational view of cartography. *The American Cartographer*, 6(2):101–106, 1979.

[vOV91] P. van Oosterom and T. Vijlbrief. Building a GIS on top of the open DBMS Postgres. In *Proceedings of EGIS'91, Brussels, Belgium*, pages 775–787, April 1991.

[WC96] J. Widom and S. Ceri. *Active Database Systems.* Morgan Kaufmann Publishers, 1996.

[ZG91] Q. Zhou and B.J. Garner. On the integration of GIS and Remotely sensed data: towards an integrated system to handle the large volume of spatial data. In *Proceedings of the Second Symposium on Advances in Spatial Databases*, pages 63–72, August 1991.

This article was processed using the LaTeX macro package with LLNCS style