A Scientific Workflow Approach to Distributed Geospatial Data Processing using Web Services

Efrat Frank[†], Ilkay Altintas[†], Jianting Zhang[‡], Bertram Ludäscher[†,§], Deana Pennington[‡], William Michener[‡]
[†]San Diego Supercomputer Center, [‡]University of New Mexico, [§]University of California Davis
{*efrat,altintas*}*@sdsc.edu,* {*jzhang,dpennington,wmichener*}*@lternet.edu, ludaesch@ucdavis.edu*

Abstract

Geospatial analyses of distributed data from surveys and sensors are often stored and managed in diverse regional, national and global repositories. The nature of scientific processes requires composition of these resources in a meaningful order to solve a specific geoscience problem. These tasks can be viewed as scientific workflows. Web based interfaces allow access to remote data and tools, and enable running computational experiments using different online resources. However, it requires manual processing to combine multiple resources in pipelines and scientists still need IT experts to automate their large-scale scientific workflows. The challenging problem is how to enable the scientists to harvest online data and models for designing and executing their experiments in a seamless manner. A solution becomes feasible by the introduction of Web and Grid Services in a variety of scientific domains. These services can be discovered and composed through generic visual interfaces and scientific workflow tools. For this purpose, we present a complete framework for creating, registering, discovering, composing and executing Web Services to support online science.

1. Introduction

Geospatial analytical functionality is essential to environmental modeling. As large scale distributed geospatial data is available, software reuse and sharing becomes more and more important in integrated environmental modeling, experimenting and analysis. The Service Oriented Architecture (SOA) allows cooperation of data and process components among different organizational units and supports reusability and interoperability of components on the Web, thus increasing the efficiency of assembly and decreasing the cost of development.

Due to the distributed nature of geospatial data, data obtained from surveys and sensors are often stored and managed in regional, national and even global repositories. This enables multi-variable and multi-scale environmental modeling at a global scope and improves our understanding of the environment. However, as most legacy systems are designed for modeling a single environmental variable at a particular region, it has become rather challenging to allow domain scientists to perform experiments and forecasts in a seamless manner by harvesting and coordinating these distributed geospatial data and environmental models.

In recent years, the need for adaptable interfaces and tools for accessing scientific data and executing complex analyses on the retrieved data has risen in a variety of disciplines (e.g., geology, chemistry, biology, ecology). Such analyses can be modeled as scientific workflows in which the flow of data from one analytical step to another is described in a formal workflow language. While traditional business workflows are oriented towards document processing, task management and control-flow, scientific workflows typically are data- and/or compute-intensive, dataflow-oriented, and often involve data transformations, analysis, and simulations. Kepler [11, 37] is a system for design, discovery, execution and deployment of scientific workflows from different scientific domains. In this paper, we propose to use the Kepler scientific workflow system to compose geospatial services for environmental modeling. To the best of our knowledge, there exists no previous work on utilizing scientific workflows for geospatial analysis and environmental modeling by composing and executing Web Services in a systematic manner.

We present an approach to provide uniform access to the vast amount of highly heterogeneous services. These services and the related metadata are archived using extensive storage capabilities through registries, and the services are discovered using a metadata description of their operation. We use the Kepler workflow system to demonstrate the registration and discovery process of services within repositories.

We further describe how within our framework, services can be composed into scientific workflows and executed to perform scientific tasks. Workflows can also be stored in

repositories and shared between scientists. The workflow execution can be monitored to detect and recover from failures, to capture intermediate and end results of the process for data provenance, and to log process information and save execution logs. A challenging issue is to provide a web based access for viewing, executing and sharing scientific workflows and deploying scientific workflows as a new service that can be applicable to other applications.

In the past decade, there has been a growing interest in scientific workflows to facilitate performing scientific experiments, as witnessed by a number of recent academic and commercial systems that have been developed. Among the academic scientific workflow system are Taverna [21] of the ^{my}Grid project, which is targeted towards providing an interface to performing biological experiments, and the Triana (GridLab) [22] problem solving environment for providing a user interface for composing scientific applications. However, neither is focused on geospatial data discovery and analysis. Within the geospatial data processing field, two leading commercial software products provide workflow composition and execution functionality, particularly, ESRI's ArcGIS ModelBuilder [7] and ERDAS Imaging's Spatial Modeler [6]. However, neither system provides support for distributed computing nor Web Services invocation. Furthermore, they enable only a sequential execution mode on a single machine. We propose Kepler as a scientific workflow system for enabling geospatial data discovery, composition and execution under a parallel model of computation.

The rest of the paper is organized as follows. In section 2, we provide a brief overview of distributed geospatial data processing and propose to use Web Services as the building blocks of distributed geospatial data processing within a scientific workflow system. Section 3 introduces the Kepler scientific workflow system's Web Services framework. The Kepler overall architecture is discussed in Sections 4, 5 and 6. Section 7 presents a running example to illustrate using Kepler to support environmental modeling. Finally, in Section 8, we provide a summary and future work directions. Although the focus of this paper is towards geospatial data processing, the solutions are applicable to other domains as well.

2. Distributed Geospatial Data Processing

In the early days of computer-aided environmental modeling, geospatial data and process sharing between different machines was available only through manually copying it using mediums, such as floppy disks or CDs. With the development of computer networks, much of this work can be automated using tools and scripts. However, this approach is still inherently labor intensive and requires considerable human interactions which is both inefficient and error prone.

During the past few years, major Geographical Information System (GIS) software vendors (e.g., ESRI [8]) expanded their software functionality to provide distributed geospatial data management using mainstream Database Management Systems (DBMS) to support computation remotely in a computer network environment. Furthermore, major database vendors (e.g., Oracle [14] have added geospatial data handling modules to their products by using the Object-Relational DBMS (ORDBMS) to provide distributed geospatial data management as well. However, there are several limitations to using a pure vendor-specific DBMS approach to distributed geospatial data processing for environmental modeling, such as cost of ownership, technology complexity and interoperability. More importantly, though database systems are naturally suited for querying/filtering using spatial indexing, they provide poor support for spatial transformations. Spatial transformations are required for transforming between data types or geospatial data values. For example, when applying Convex Hull algorithms, a point dataset is transformed into a polygon dataset. Another example is of overlaying two polygon datasets where the output has different values than the inputs. Nowadays, major commercial database systems, for instance, Oracle, are beginning to offer some spatial transformations capabilities. However, the underlying ORDBMS model still makes it difficult to apply them to environmental data, since the data is usually unstructured or semi-structured. Therefore, a pure distributed spatial DBMS approach based on SQL-like queries for environmental modeling is undesirable if not infeasible. On the other hand, although Web GIS [35] has put major efforts towards distributed geospatial data processing, adopting the client/server architecture, they are mostly restricted to visualization capabilities, offering little support for complex queryies and analysis.

Furthermore, rendering geospatial data at client side in the form of images or Java/COM object makes the integration and reuse of geospatial data very inefficient. Although using XML as the communication protocol has been proposed for geospatial data integration purposes [39], distributed geospatial data processing that involves data transformation has hardly been investigated.

SOA provides a publishing interface to data and tools using the Web Service Definition Language (WSDL) [24]. WSDL is an XML based language which enables client or program interaction with a Web Service. Web Services communicate through the Simple Object Access Protocol (SOAP) [17]; an XML based protocol understandable to both machine and human. Both WSDL and SOAP are platform independent and have a wide range of support. From a geospatial data processing perspective, Web GIS is a special case of Web Services where raw data is transformed into a browser compliant format. The SOA allows geospa-

tial data processing methods to be exposed to the Web and it is found to be very useful for environmental modeling in a distributed environment. Thus, we consider the SOA a promising paradigm in distributed geospatial data processing.

We envision that an open architecture is vital for newly emerging integrated and distributed environmental modeling. The architecture should support (1) both flat data (such as operation system files), semi-structured and structured data (such as databases), (2) legacy models written in traditional languages or scripts, and (3) interactive and automatic executions of environmental models in the form of scientific workflows. We believe that using Web Services as the building blocks for geospatial data processing within a scientific workflow system fulfills the above requirements.

In this paper, we propose publishing geospatial data and processes as Web Services and composing them using a scientific workflow approach. In the next section we present the Kepler scientific workflow system as our Web Service composition and execution framework to achieve distributed environmental modeling.

3. Kepler Web Service Framework

Kepler [11] is a system for the design and execution of scientific workflows. It is built on top of the PtolemyII system, a modeling and design tool for assembling concurrent components by means of various models of computation [15]. Kepler is an extensible open source scientific workflow system that provides scientists with a graphical user interface to register and discover resources, and the capability to interactively design and execute scientific workflows using emerging Web and Grid-based technologies to distributed computations.

Kepler is unique in that it seamlessly combines high-level workflow design with execution and runtime interaction, access to local and remote data, and local and remote service invocation along with a built-in concurrency control and job scheduling mechanism. Other unique features are inherited from the underlying PtolemyII system, e.g., the ability to combine different models of computations in a single scientific workflow.

Computational units in Kepler are called *actors*, which are reusable components that communicate with each other via input and output ports. Actors are linked to each other to compose a *scientific workflow*. The workflow execution is orchestrated by a *director* that provides the model of computation, that is, scheduling components interaction. In this paper we explain how the Kepler environment can be utilized to discover, compose and execute geospatial data processing workflows for environmental modeling, most essentially using a generic Web Service invocation component.

Several generic Web (and Grid) services actors have been implemented in Kepler that serve as clients for accessing distributed resources within Kepler workflows. Specifically, the WebService actor provides the user with a simple plug-in mechanism to execute any WSDL-defined Web Service. The user can instantiate the generic Web Service actor by providing the WSDL URL and choosing the desired Web Service operation. The actor then automatically specializes itself and adds ports with the inputs and outputs as described by the WSDL. The so instantiated actor acts as a proxy for the Web Service being executed and links to the other actors through its ports. Using this component, any application that can be deployed as a remote service, can be used as a Kepler component.

Other features of the Kepler framework to support Web Service execution are shown in Figure 1 and are discussed in the next sections. The figure depicts the Kepler overall architecture for facilitating scientific experiments. The sequence of events involved in performing and analyzing a scientific experiment are as follows. The user or provider publishes scientific datasets and processes. The user can then access distributed scientific resources by searching and harvesting them. Discovered components can be composed to a scientific workflow and may also be registered within the system either as a local component or within domain repositories. The system provides several features for monitoring workflows execution, such as, failure recovery, data and process provenance and post execution processing. Another functionality that is currently under research and development is the deployment of a scientific workflow as a new remote service.

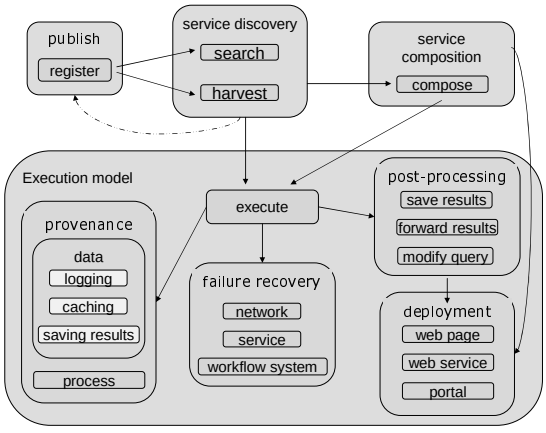


Figure 1. Life-cycle of Kepler Web Services.

4. Registries

For the past decades, considerable efforts have been made towards standardization of geospatial data formats, such as the International Organization for Standardization Technical Committee 211 (ISO/TC211) and the Open Geospatial Consortium (OGC) abstract and implementation standards [34, 36]. The acceptance levels, however, vary greatly. For example, while some systems push towards geospatial databases for data storage and manipulation processing, others still use flat files with undocumented data formats. Furthermore, even systems designed for distributed computation, may use different approaches, such as DCOM/DotNet, J2EE or CORBA [38]. These issues make the integration process a non-trivial task.

As the Geographical Markup Language (GML) [36] is soon to become the ISO International Standard (IS) 19136 and major GIS vendors provide support for it, it is expected that GML can bridge between the various data formats. Thus, we propose to use the GML data format and the SOA in distributed geospatial data processing. In order to efficiently access distributed data and process services, we use registries that are accessible through the Kepler system.

Web Services registries allow publishing services in remote repositories based on the service operation description. The purpose of registering services is to facilitate their discovery and provide methods of their execution. In the Kepler system, services are registered using domain ontologies and can be discovered either syntactically, using a text based search, or semantically by querying over concepts in the related domain ontologies. The main goal is provide a unified framework for handling registries. In the following sections we elaborate on service registration and discovery within our framework.

4.1. Registration

We distinguish between two types of repositories, a Universal Description Discovery and Integration (UDDI) [23] approach, and domain based repositories. The domain based repositories are pluggable repositories that are specific to a scientific domain. Within the domain based repositories, services are semantically registered with concepts from domain ontologies and are discovered by querying through reasoning with concepts in the ontology.

A service in our framework can be a *process service* to perform an operation (e.g., a Kepler actor), or a *data service* to query over a dataset. In the example in Section 7, datasets of different formats are accessed through query services that return a GML representation of the original data. Those services are stored in the repositories as data services. The processes used in the example are GRASS [9] geospatial analytical functions, wrapped as Web Services to provide a unified access framework. These services are registered in a domain repository as process services. The registration process is as follows:

Mapping Services to Ontologies For semantically discovering a service, it first needs to be linked to one or more concepts in an ontology(-ies). Ontologies are selected from a list of ontologies registered in the system. The service is mapped to an equation over selected concepts from the selected ontologies. Two types of registration approaches are currently supported: *item level* and *item-detail level* registration. When registering a process or a Web Service, the whole unit (or item) is registered to concepts in the ontologies, that is, an item level registration. Registering a database, allows the user either to register the whole unit to concepts or to map one or several columns of the database schema with concepts, that is, item-detail level registration. A service registered to a concept is automatically registered to all of its super and equivalent concepts in the ontology hierarchy through a reasoning service.

Annotating Services with Metadata Services are also annotated to provide general information, such as, the service creator, a description of the service, keywords, etc.. For databases and shapefiles, the schema is extracted and the user can provide further details about each column. The database schema is cached to facilitate querying of the dataset through graphical interfaces. Currently, several metadata repositories are supported in Kepler: the Geosciences Network (GEON) [13] catalog, where services are annotated using Digital Library for Earth System Education (DLESE) ADN metadata [3] and its schema saved in a PostgreSQL database, the Science Environment for Ecological Knowledge (SEEK) [12] Metacat which is the Knowledge Network for Biocomplexity (KNB) repository that stores Ecological Metadata Language documents (EML) [5], the Storage Resource Broker (SRB) [18], which is an operating system like storage mechanism that provides additional metadata functionality on top of the file system organization, and the Distributed Generic Information Retrieval (DiGIR) [4] for UDDI registered data.

4.2. Discovery

Two approaches are supported for discovery of services: searching within domain based repositories and harvesting Web Services from a UDDI or a web based repository. Both approaches and the reusability of discovered components are discussed below.

4.2.1. Searching Domain-Based Repositories The domain-based repositories are pluggable repositories that are accessible to the system. To make a distinction between the three different (not necessarily disjoint) types of services mentioned above, each is accessible in a dif-

ferent tab. A process tab is used for listing and searching for actors, directors and other workflow aiding tools. A data tab is for services for querying local and remote datasets, such as relational databases, XML, flat files and shapefiles. A Web Services tab refers to Web (or Grid) services, identified by a URI. Data and Web Services can also be classified as process service under the process tab, using generic database and Web Service actors that can be instantiated to link to a specific location.

The search process, which is similar in all three tabs, is as follows: A search can be either syntactic, that is, a text based search by the services names, or semantic by issuing a query against semantic information stored in the registries. A semantic search would result in all matching services satisfying the equivalent and 'isa' relations of the query (an expression over concept). The search returns a service name, id or a URL which is then processed depending on the type of the service. Below we elaborate on the different processing of the search results that is performed in each tab.

Actors Tab The actors tab lists process services within the Kepler system. The search within this tab would return an actor name which would extract its java based content to perform a specific task once dragged onto the workflow canvas. The services in this tab can be local or remote.

Data Tab Searching the data tab for a dataset returns a dataset id wrapped as a Kepler component. This component has information to access the discovered dataset depending on its type. Once dragged onto the canvas, the component is populated with necessary schema information and input and output ports. Accessible through these components is a query builder interface that accepts the dataset schema and allows the user to easily query it.

Service Tab The Service tab search returns either an id to a specific Web Service method or a WSDL URL to access all methods of a Web Service. The search results are wrapped as components that serve as a client to the specific Web Service methods. All input and output ports are automatically instantiated as specified by the service requirements.

4.2.2. Web Services Harvester Kepler provides a Web Service harvester capability for importing Web Services from a repository. This feature was developed for conveniently plugging in a whole set of (possibly related) services. The Web Service descriptions to be imported can be harvested from a simple web page or retrieved from a structured repository like a UDDI or WSIF [25]. Once imported, the Web Services are saved as actors. These actors can then be reused in different scientific workflows.

4.2.3. Reusability Within the Kepler system, components are workflow independent, that is, components may be applicable in acting in more than the single scientific experiment for which they were initially harvested. The system provides a capability to save discovered components either as local components or in a specific domain repository by registering the component as described in section 4.1.

5. Service composition

Using Web Services within a workflow environment gives the ability to describe a process to facilitate modifying, executing and monitoring. Combining several Web Services together provides coordination of resources that are not necessarily subject to a centralized control to deliver a single procedure. Current Web Services composition frameworks, for example BPEL4WS [1], describe composing multiple Web Services to provide a composite Web Service. However their main focus is on the syntactic level of composition and thus do not allow automatic composition of Web Services. In the following section we elaborate on Web Service composition languages and discuss on tools for enhancing automatic composition in the Kepler framework.

5.1. Web Service Composition Languages

The Workflow Management Coalition (WfMC) [26] defines a workflow as "the computerized facilitation or automation of a business process, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". A Workflow Management System (WfMS) is "a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic. It consists of a modeling as well as a runtime component". Although the above definitions refer to "business processes", they also apply to scientific workflows.

Web services coordination requires long running interactions, driven by explicit process models [33]. The logic of a composite Web Service can be captured by a modeling language. Many such languages have recently emerged, such as, BPEL4WS [1], BPML [2], WSCI [31], and XPDL [28]. Among the scientific workflows languages that support Web Service composition are Kepler/PtolemyII's *MoML*, Taverna's Scufl, and Triana's TaskGraph. BPEL4WS and MoML are further discussed below.

In the business data processing field, Business Process Execution Language for Web Services (BPEL4WS) [1] is now playing an important role in business process modeling. Built on top of IBM's graph oriented Web Services Flow Language (WSFL) and Microsoft's procedural languages Web Services for Business Process Design (XLANG), BPEL4WS is intended to provide a conve-

nient language to express the majority of workflow processes, supporting modeling of how multiple Web Services could be composed to provide a new composite Web Service.

A BPEL4WS workflow is designed to act as a Web Service and can easily interact with other Web Services to compose a new service. It describes interoperability of services that are consuming and producing messages described in WSDL port types. However, the current infrastructure focuses entirely on syntactic interoperability, thus limiting the automatic composition of Web Services functionality. In the next section we discuss the Kepler framework extensions to enhance automatic Web Services composition.

Kepler uses the Modeling Markup Language (MoML), inherited from the underlying PtolemyII system, as its workflow representation language. MoML is an extensible XML modeling language that can be easily shared between scientists. It is intended for specifying interconnection and hierarchies between workflow components. MoML is not solely intended for Web Services composition. It is object-based; each component's instance (actor) is defined by its java object, and thus can be used to represent interactions of various local and remote processes identified by their java classes. Furthermore, any BPEL4WS workflow can be represented as a MoML workflow using the generic Web Service actor and other intermediate components. In the future, we plan to extend Kepler with plugins to explicitly accommodate external workflow languages representations.

5.2. Enhancing Automatic Composition

Kepler provides an extensible type system for defining the type of inputs and outputs that can be passed through actors ports. The system can perform both a design and run time type checking on the flow of data between workflow components, using a built-in type lattice, to guarantee compatibility of datatypes during workflow composition and execution. However, due to the distributed nature of Web Services, often times Web Services are not designed to fit. Thus, data transformations between the outputs of previous steps and inputs of subsequent steps are usually required. In this section, we discuss extension to Kepler to support syntactic and semantic interoperability.

5.2.1. Syntactic and Datatype Conversions As Web Services are not subordinated to a single authority, data transformation components are required to match the outputs of previous steps with the inputs of subsequent steps within a workflow. Kepler's type checking mechanism ensures that the input type of a subsequent actor is at least equal to or higher than the output of a previous actor, within the Kepler type lattice. Aside from built-in transformations, several specialized data transformation components have been implemented. Among these components are various datatype conversion components, an XSLT-based transformer [30], and generic XQuery [29] and XPath [27] components. Kepler also provides a browser user interface functionality that offers the user with a graphical user friendly interface to specify inputs.

The Kepler datatype conversion actor is a type polymorphic component that accepts any input type, within the Kepler type signature, and a desired output type, and attempts to translate between the designated types. An example of a group of datatype conversion actors is a set of XML actors that translate between XML schemas and their string representations. Other data transformation components such as XSLT and XQuery provide syntactic transformations to produce a specific required input format.

In the example in Section 7, transformation services are used to convert various geospatial data formats into GML. A streaming approach is then used to process GML data over geospatial Web Services which makes their invocations more convenient and robust.

5.2.2. Semantic Conversions Often times closely related Web Services produce or consume closely related data (either syntactically or semantically). For example, when handling geological data, we often run into this when integrating between datasets using different geologic standards (e.g., different geologic time scales, different rock classification hierarchies, etc.). These inconsistencies can be solved by using an articulation ontology mapping between one standard to another. Another example of semantically related, though inconsistent, data is given in the following scenario: some Web Services may produce as an output a URI handle to access the generated data while others may expect the actual data as input, further knowledge is required to determine the necessity of a dereferencing mechanism. The above mentioned correlations may be identified using a domain knowledge represented by an ontology. The solution to mediating between such closely related, although incompatible, subsequent Web Services inputs and outputs is known as *semantic data transformations* [32] and is currently being researched. Below we discuss future extension within Kepler's Web Services framework to support semantic mediation.

As Web Services are being registered in the repositories they are being annotated with concepts from a domain ontology. Future plans are to annotate each input and output port of a service with semantic types. These types will be used when trying to resolve mismatching subsequent input and output types while composing several services together. Currently, coordinating between incompatible datatypes is supported by using intermediate components wrapped as Web Services to access and reason with domain ontologies. Future work is towards extending the type system to sup-

port direct semantic interoperability between input and output types of subsequent components and for semantic discovery of services by the expected conceptual inputs and outputs types.

6. Web Service-Based Workflow Execution and Deployment

6.1. Monitoring and Failure Recovery

Since service-based workflows depend on distributed computational resources under diverse controlling authorities, they are exposed to a high risk of component failure, including failures in computational platforms, network, application services or the workflow system itself. In order to eliminate these situations, scientific workflows with distributed components need closer monitoring and alternative plans to recover. The monitoring requirements vary for distributed workflows having different characteristics. These range from service monitoring to workflow process execution monitoring and differ for long and short running workflows. Short running workflows generally do not require a job database and the execution doesn't need detaching from the workflow as the execution time is relatively shorter. The long running workflows generally require some sort of batch processing and require tracing the status of jobs in a job database to be monitored from time to time.

Monitoring of actors and tools in our framework is designed to be scalable depending on user needs. We propose three different levels of monitoring, namely, light, standard, and heavy. In the standard monitoring level, the user is notified only if an actor fails to execute. The standard workflow execution monitoring is done via a monitoring subsystem that interacts with the director. It notifies the user about the failures immediately, and keeps track of the intermediate results to be reported to the user after the workflow execution finishes. The light monitoring system is one that watches the execution but does not notify the user about failures until the workflow has finished or stalled. The heavy monitoring verbosely reports every communication between the workflow entities and also notifies the user about failures immediately.

6.1.1. Network Failures The distributed nature of service-based workflows makes them vulnerable to network failures, often resulting in a failure caused by events like losing network connection, failure of a SOAP server, or simply migration of the server from one domain to another. Many of these issues fall into the domain of system and network administrators, who must design an infrastructure to provide redundant components. In this work, we address only the parts that are under our control.

In our framework, we choose to keep track of the failed services and the parts of the workflow which depend on them, and to retry to connect to a failed resource(s) or service after a certain amount of time, or when the connection can be reestablished in case of lost network connection.

6.1.2. Service Failures The majority of failures in service-based workflows happen due to service failures and bad input. The former happens due to faulty WSDL file or a changed service location, whereas the latter happens when the service is down or obsolete, making the workflow component that runs the service malfunction. A solution to this is to register alternative Web Services that can be used interchangeably in cases of failures. Alternative resources may be either found in a service registry, or may be hard-coded into workflows. The decision on where to find them depends on the fail-over policy/strategy of a particular workflow system. Also, the discovered/harvested Web Services needs to be updated by the user before being used in a specific workflow. The suggested framework satisfies these requirements at component level. This will later be reflected to the behavior of the execution engine (a separate director) and will be handled at the system level.

6.1.3. Workflow System Failures These are the type of errors that are caused by a missing parameter in a workflow like a missing username and/or password for a Web Service. The workflow system needs to detect these errors in advance and notify the user before the workflow execution begins. Our framework currently doesn't detect these errors prior to runtime. Plans for generating a new director also include identifying possible errors during workflow design and as the execution starts.

6.2. Data and Process Provenance

In scientific applications it is often necessary to keep track of data and processes that were used to produce the results of a computational experiment or scientific workflow, in particular to facilitate reproducibility. This provenance information can be associated with a result dataset or workflow run, effectively providing an execution trace of certain crucial provenance information. Logging services can be customized and integrated into a scientific workflow system for this purpose. Such services provide interfaces to modify log filters and monitor and create views over previous logs.

Scientific workflows may require user interaction at runtime, e.g., to determine which data subsets should be routed through which of several alternate paths of the workflow, or for computational steering. Workflow engines already maintain information on intermediate steps and execution details of processes. The challenge is to display that infor-

mation in a way that will satisfy the needs of different users, with several detail levels and provisions for a variety of different publishing methods.

We propose an optional interface that can be plugged in to the workflow. This plug-in component gives the user a set of customization options on where to archive and deploy the acquired information, in which format to publish it, the specifics of the information to collect (e.g. end/intermediate results, images and graphics, database rows/ids, metadata, etc.), and the display methods. Once complete, this will also be included as a service in the existing service repository.

6.3. Workflow Deployment

An important requirement of a workflow system is to provide a detached execution model to allow users to execute scientific workflows from within other environments than the local application. Currently, Kepler provides users with functionality to view and execute workflows on a web page through applets, however, this approach limits the execution capabilities to the applets framework restrictions. Another approach that was examined was using Java Web Start to execute workflows in a browser environment. Web Start is an efficient method to deploy production workflows and provide automatic updates on them and was used to deploy some Kepler workflows. Yet, this approach was not pursued since it requires certain signatures for packaged archives that do not necessarily match the providers' signatures. The topic of workflow deployment is currently being researched. In this section we discuss the significance of workflow deployment and present several models for achieving this goal.

6.3.1. Motivation The Web Service framework is targeted at executing processes that are deployed on remote machines. Deploying Kepler workflows as Web Services would provide a detached execution mode to execute scientific workflows. A detached execution mode gives scientists means to execute workflows within other service-based applications and also allows them to easily share experiments' results with each other. Furthermore, it would provide a *detachable* execution mode that allows monitoring of long running workflows and compute-intensive tasks using the workflow control engine to run in the background on a remote server, without necessarily staying connected to a client application.

Currently, the Web Service framework provides no user interaction support during remote *workflow service* execution, nor are GUI displays available to deploy workflows or parts of workflows automatically as web services. Kepler comes with several user friendly display capabilities and interactive shells to assist user interactions, thus Kepler workflows cannot be intuitively deployed as Web Services and additional steps are required. In the next section we elaborate on several deployment models that try to overcome the above mentioned obstacles.

6.3.2. Deployment Models

No user interaction The first model assumes that the workflow to be deployed as a web service contains no GUI components and no user interactions (I/O components). The deployment of such a workflow requires the deployment of a small execution engine with the workflow. Kepler provides a command line execution mode that can be easily deployed by wrapping it with the workflow description as a Web Service.

Sub-workflow deployment A GUI actor cannot take part in a Web Service execution as it opens a separate display window. The following model allows deployment of sub-workflows that contain no I/O components, thus parts of the workflow become available to run on the server side. A generic Web Service actor can later be used to invoke the deployed sub-workflows. Two deployment options are planned, a user triggered deployment and deployment of maximum connected components.

In a *user triggered deployment*, the user selects several connected components using the Kepler graphical user interface, *Vergil*, and requests to deploy them. The system verifies that the sub-workflow contains no GUI components, and deploys it on the server side.

In *deployment of maximum connected components*, the workflow is examined to detect maximum connected components (sub-workflows) that do not involve any I/O functions. These components are deployed as Web Services. The deployment procedure is as follows:

1. The current workflow descriptionin, MoML, is first translated into a relational form that allows reasoning over components' relations within the workflow.
2. The relational representation is traversed using graph traversal algorithms to detect maximum connected components that contain no I/O components.
3. Each sub-workflow identified in the previous step is deployed as a Web Service. The workflow is modified to use the Web Service actor in place of the sub-workflows.

The deployment process is transparent to the user. The user assumes to run the original workflow while in the background the Web Service actor replaces deployed components.

Portal Deployment Deploying Kepler workflows in an existing portal is currently being examined. We are currently investigating methods to use the GridSphere portal framework [19] along with a possible distributed client/server Ke-

pler architecture to execute a workflow from within a portlet or to execute the Kepler system on the server side from a portal environment.

7. Geospatial Data Processing Example

The following example in species occurrences analysis is used to illustrate geospatial data processing within the Kepler system. The goal is to find all the occurrences of species A that are within the intersection of the convex hulls of the occurrences of species B and species C. We assume that the occurrences datasets (point data) are stored in three different formats: species A data is stored in a flat file, species B data is stored and managed by SQL Server 2000 and species C data is stored and managed by Oracle 10g with spatial capabilities. We further assume that these three datasets are accessible through Web Services which return a GML representation of the data. Three geospatial functions are used to process the query: GRASS' Convex Hull, Oracle Spatial's Polygons Intersection and a Java Point in Polygon algorithm. These functions are wrapped as Web Services to provide a uniform, domain independent access. Finally, a visualization component is used to display GML documents. This component wraps GeoTools' [10] GML displayer as a Kepler actor. All of these components are registered within a geological repository and are discovered, composed and executed within the Kepler scientific workflow system.

Figure 2 provides a snapshot of the geospatial data processing workflow. The workflow composition process that was applied in creating this workflow is as follows: The user first searches for the desired data and process services. The data services are discovered within the data tab while the process services are discovered within the service tab. Discovered services appear in top left panel of the Kepler graphical user interface and can be dragged and dropped onto the workflow canvas to perform within a scientific workflow. The services are linked to one another from their visual ports using the GUI. The semantic data transformations in this example are transparent to the user. The datasets are automatically transformed into a GML format while being accessed, using the discovered data services, therefore, no additional intermediate components are required. As for the process services, those were initially designed to consume and produce GML document strings, and thus require no further format integration processing.

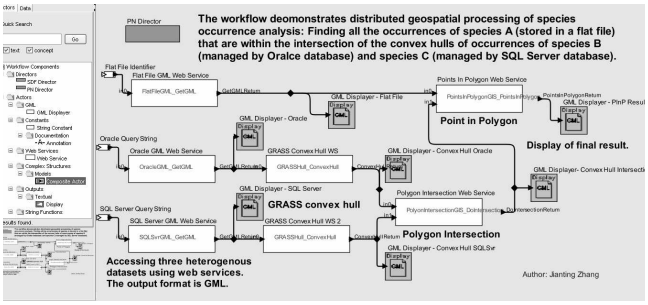


Figure 2. Modeling species occurrences.

As shown in Figure 2, accessing the datasets and the two Convex Hull operations can be done concurrently while the Point in Polygon and Polygon Intersection execute sequentially. Such an execution is feasible in Kepler through a *Process Network director*, (PN) [20], which schedules the workflow execution to a parallel mode (when possible) by creating a separate execution thread for each actor.

GML display actors are used to provide a visualized display of the resulting datasets of each processing step. This actor may also be replaced with a string display actor to display the resulting GML content, or a file writer to store the computation result. Kepler also provides support (as mentioned above) for execution monitoring, such as, a timeout mechanism for connecting to a Web Service, exceptions handling, and enabling to continue halted executions from a certain checkpoint by caching components' computations. The interactive workflow system essentially provides users with a visual programming environment that can effectively harvest, compose, execute and monitor distributed geospatial processes for environmental modeling.

8. Discussion and Future Work

In this paper, we have described a framework for service-based workflows and illustrated the application of it to distributed geospatial data processing for environmental modeling. The Kepler scientific workflow system provides a convenient interface for discovering and harvesting services, and interactively composing and executing them with support for execution monitoring and data provenance.

We demonstrate the feasibility of the proposed approach for geospatial data processing with a simple running example within the Kepler system. The example emphasizes Kepler's strength to facilitate environmental modeling. By publishing geospatial data and processes as Web Services, a unified framework is achieved for accessing various formats and systems. To the best of our knowledge, there exists no previous work on experimenting with geospatial data through scientific workflows, using Web Services as building blocks. Although the focus of this example is towards environmental modeling, the proposed solutions have already been utilized within other scientific domains that uses the Kepler system's web service framework.

As for future work, aside from the inline mentioned extensions to Kepler to provide a better support for automatic

composition and deployment of scientific workflows, from a geospatial perspective, we would like to develop a client application to facilitate publishing of datasets as GMLs, including solving efficiency issues when processing very large datasets.

We are also interested in extending Kepler's web service framework in a direction that allows efficient third-party data transfers, has specialized directors (computation engines) for optimized distributed execution for service-based scientific workflows, and provides a role-based Kepler access framework that allows users to create ad-hoc usage groups in order to enable them share their data and processing services through a unified infrastructure.

Acknowledgments

Kepler includes contributors from SEEK [12], SDM Center-SPA [16], Ptolemy II [15] and Geon [13]. Work supported by NSF ITRs 022567 (SEEK), 0225673 (GEON), DOE DE-FC02-01ER25486 (SciDAC/SDM), DARPA F33615-00-C-1703 (Ptolemy), and DARPA N00017-03-1-0090. Special thanks to Kai Lin and the rest of the GEON and SEEK Kepler developers for providing the registration infrastructure and advising in Kepler's development stages.

References

[1] Business Process Execution Language for Web Services (BPEL4WS), http://www-128.ibm.com/developerworks/library/ws-bpel/.

[2] Business Process Modeling Language (BPML), www.bpmi.org.

[3] Digital Library for Earth System Education (DLESE), www.dlese.org.

[4] Distributed Generic Information Retrieval (DiGIR), digir.sourceforge.net.

[5] Ecological Metadata Language (EML), knb.ecoinformatics.org/software/eml/.

[6] ERDAS Spatial Modeler, http://www2.erdas.com/ support-site/procedures/imagine/spatial_model/spatial_model.html.

[7] ESRI Arcview ModelBuilder, http://www.esri.com/library/whitepapers/pdfs/model_bldravsa2.pdf.

[8] ESRI, http://www.esri.com.

[9] Geographic Resources Analysis Support System (GRASS), http://grass.itc.it/index.php.

[10] Geotools Lite, http://sourceforge.net/projects/geotools/.

[11] Kepler: An Extensible System for Scientific Workflows, http://kepler.ecoinformatics.org.

[12] NSF/ITR: Enabling the Science Environment for Ecological Knowledge (SEEK), www.seek.ecoinformatics.org.

[13] NSF/ITR: GEON: A Research Project to Creat Cyberinfrastructure for the Geosciences, www.geongrid.org.

[14] ORACLE Spatial, http://www.oracle.com/technology/products/spatial/index.html.

[15] PtolemyII, http://ptolemy.eecs.berkeley.edu/ptolemyII/.

[16] Scientific Data Managment Center (SDM), http://sdm.lbl.gov.

[17] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap/.

[18] Storage Resource Broker (SRB), http://www.sdsc.edu/srb/.

[19] The GridSphere Project, www.gridsphere.org/gridsphere/gridsphere/.

[20] The Process Networks (PN) Domain, http://ptolemy.eecs.berkeley.edu/publications/papers/99/HMAD/html/pn.html.

[21] The Taverna Project, http://taverna.sf.net.

[22] The Triana Project, http://www.triana.co.uk.

[23] Universal Description Discovery and Integration (UDDI), http://www.uddi.org.

[24] Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl.

[25] Web Services Invocation Framework (WSIF), ws.apache.org/wsif/.

[26] Workflow Management Coalition (WfMC), http://www.wfmc.org.

[27] XML Path Language (XPath): http://www.w3.org/TR/xpath.

[28] XML Process Definition Language (XPDL), www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf.

[29] XQuery 1.0: An XML Query Language": http://www.w3.org/TR/xquery/.

[30] XSL Transformations (XSLT): http://www.w3.org/TR/xslt.

[31] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek. Web Service Choreography Interface (WSCI) 1.0, http://www.w3.org/TR/2002/NOTE-wsci-20020808/.

[32] S. Bowers and B. Ludäscher. An Ontology-Driven Framework for Data Transformation in Scientific Workflows. *Data Integratin in Life Sciences (DILS)*, 2994:1–16, 2004.

[33] W. V. der Aalst. Don't Go With the Flow, Web Services Composition Standards Exposed. *IEEE Intelligent Systems*, 18(1), Jan/Feb 2003.

[34] D. Green and T. Bossomaier. *Online GIS and Spatial Metadata*. Taylor & Francis, 2002.

[35] A. R. Hardie. The Development and Present State of Web-GIS. *Cartography*, 27(2):11–26, 1998.

[36] W. Kresse and K. Fadaie. *ISO Standards for Geographic Information.* Springer, 2004.

[37] B. Ludaescher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, to appear 2005.

[38] Z.-R. Peng and M.-H. Tsou. *Internet GIS : Distributed Geographic Information Services for the Internet and Wireless Networks.* Wiley, 2003.

[39] J. Zhang, M. Javed, A. Shaheen, and L. Gruenwald. Prototype for wrapping and visualizing geo-referenced data in a distributed environment using XML technology. *ACM-GIS*, pages 27–32, 2000.