Use of grid computing for modeling virtual geospatial products

Aijun Chen, Liping Di, Yaxing Wei, Yuqi Bai & Yang Liu

Center for Spatial Information Science and Systems, George Mason University, Greenbelt, MD, USA

Available online: 24 Jul 2009

To cite this article: Aijun Chen, Liping Di, Yaxing Wei, Yuqi Bai & Yang Liu (2009): Use of grid computing for modeling virtual geospatial products, International Journal of Geographical Information Science, 23:5, 581-604

To link to this article: http://dx.doi.org/10.1080/13658810902733666

International Journal of Geographical Information Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
Use of grid computing for modeling virtual geospatial products

AIJUN CHEN*, LIPING DI, YAXING WEI, YUQI BAI and YANG LIU
Center for Spatial Information Science and Systems, George Mason University, Greenbelt, MD, USA

(Received 21 July 2007; in final form 14 December 2008)

Earth science research and applications usually use Distributed Geospatial Information Processing (DGIP) services and powerful computing capabilities to extract information and knowledge from large volumes of distributed geospatial data. Conceptually, such processing can be abstracted into a logical model that utilizes geospatial domain knowledge to produce new geospatial products. Using this idea, the geo-tree concept and the proposed geospatial Abstract Information Model (AIM) have been used to develop a Grid workflow engine complying with geospatial standards and the Business Process Execution Language. Upon a user’s request, the engine generates virtual geospatial data/information/knowledge products from existing DGIP data and services. This article details how to (1) define and describe the AIM in XML format, (2) describe the process logically with an AIM, including the geospatial semantic logic, (3) conceptually describe the process of producing a particular geospatial product step by step from raw geospatial data, (4) instantiate AIM as a concrete Grid-service workflow by selecting the optimal service instances and data sets, and (5) design a Grid workflow engine to execute the concrete workflows to produce geospatial products. To verify the advantages and applicability of this Grid-enabled virtual geospatial product system, its performance is evaluated, and a sample application is provided.

Keywords: Grid computing; Abstract modeling; Virtual geospatial product; Grid service chain; Process workflow

1. Introduction

The development of Web/Internet technology has facilitated collaboration in Earth science research and applications from a local scale to the global scale (Fusco et al. 2005, Ter Linden et al. 2005, Wang and Liu 2009, Zhang and Tsou 2009). In such a complex environment, the knowledge of discipline experts plays an important role (Zhuge 2004, 2006, Ludascher et al. 2005, Babik et al. 2006, Hu and Bian 2009) in Distributed Geospatial Information Processing (DGIP) for Earth science research and applications. How to extract this knowledge and how to use the knowledge to efficiently process geospatial data for applications and decision makers remain major issues (Raskin and Pan 2005, Lutz and Klein 2006).

Conceptually, the step-by-step processing from raw data to an application-specific geospatial product forms an abstract DGIP model. This model captures expert knowledge on how to produce an application-specific product from the available raw data. To facilitate this process, we adopted the geo-object and geo-tree concepts...
The basic knowledge from the experts is abstractly defined, expressed, and provided in geo-object modules that are the foundations for constructing logical models. Then, the available geo-object modules are used to incorporate the complex logic of the application procedures into a geo-tree abstract model. This abstract model is automatically instantiated into a concrete service chain by a workflow engine, which also executes the service chain to produce useful products. The model can be considered a virtual geospatial product (VGP) that does not preexist but is available upon request. Defining a virtual product with an abstract DGIP model allows a virtual product system to quickly capture the domain knowledge of experts and express this derived knowledge in products. An Abstract Information Model (AIM) can be used to implement a DGIP system to easily leverage available distributed online resources to produce VGPs. This kind of product virtualization allows the system to offer an unlimited number of products with little effort. When a user requests a VGP, the model is first instantiated into a concrete workflow. The workflow consists of instances of the concrete services used to generate an application-specific product from distributed data sets. Conversion from a geospatial processing model to an executable workflow incorporates the user requirements for the requested product into the workflow, ensuring that the product matches these requirements.

Given the complexity of Earth science research and the large volumes of geospatial data involved, the DGIP processes are often data- and computing-intensive, requiring high-performance and high-throughput computing. For sharing geospatial resources, the Open Geospatial Consortium (OGC) has developed standards and specifications, such as the Catalogue Service/Web Profile (CS/W, Nebert and Whiteside 2005), the Web Coverage Service (WCS, Evans 2003), the Web Map Service (WMS, de la Beaujardière 2001), and the Web Feature Service (WFS, Vretanos 2002). These OGC Web services can be used to build a geoprocessing workflow (service chaining) (Friis-Christensen et al. 2009). Therefore, Grid computing (Foster et al. 2001) and OGC Web services are integrated in this study to build Grid service chaining using online computing, data resources, and geospatial standards.

This article describes the AIM concept and implementation using geo-tree, grid computing, and geospatial Web services. The objective is to effectively use existing knowledge to process geospatial data and extract new knowledge products. Section 2 reviews research relevant to the workflow engine. Section 3 introduces the geo-object and geo-tree concepts. Section 4 describes the development of a geospatial AIM and its schema and implementation based on the geo-tree concept. Section 4 also introduces a system catalog for registration of AIMs. Section 5 details how experts can use the AIM to build an application-specific Logical Processing Model (LPM), and register and retrieve it from the catalog. Section 6 describes how to instantiate an LPM into a concrete grid-service chaining workflow. Section 7 presents a Business Process Execution Language (BPEL)-compliant grid service workflow engine and describes how to use the engine to execute a concrete workflow. Section 8 describes the system architecture and implementation. A use case is introduced to evaluate performance. Conclusions are in Section 9.

2. Related work

Use of scientific workflows to implement virtual data products from existing distributed data and computing resources in Grid environment is a promising approach (Deelman et al. 2003, Amin et al. 2004, Bardeen et al. 2005, Di et al. 2005, 2006, Cybok
2006, Yu and Buyya 2006, Zhao et al. 2006, Barker and van Hemert 2008). Most representative scientific workflows have their own languages to describe and construct domain-specific workflows, but generally do not follow domain standards. The BPEL for Web Services (BPEL4WS) standard has been widely adopted for building workflows (Cybok et al. 2006, Yu and Buyya 2006, Friis-Christensen et al. 2009).

The Kepler system (Ludascher et al. 2005) is an extensible dataflow- and actor-oriented workflow system for bioinformatics, geoinformatics, and cheminformatics. It is Web Service based, not Grid service based, although it provides a few Grid functions, such as GridFTP and Globus job scheduling. Our system, in contrast, is based on an abstract model, geospatial standards, and geospatial use cases. The Kepler metadata provides semantic annotations to describe components and their inputs and outputs.

Planning for execution in Grids (PEGASUS) (Zhao et al. 2006) provides a virtual data Grid to manage, trace, communicate, and explore the creation and analysis of diverse data objects in a scalable, heterogeneous computational environment. PEGASUS has two primary components: (1) a virtual data schema for defining the objects and the relationships among the objects and (2) a virtual data system for users to construct and maintain the system in a distributed context. Zhao et al.’s (2006) research is focused on high-energy physics and astronomy and has some similarities to our research.

Deelman et al. (2003) proposed an approach for generating workflows for the Grid that describe the execution of a complex application built from individual components. The authors have developed two types of workflow generators: a concrete workflow generator and an abstract and concrete workflow generator. Both generators construct workflows for specific application domains, such as high-energy physics and geophysics, using application specific components. Users must manually discover what resources are available and where the replicated data are located.

Cybok (2006) has proposed a Grid workflow infrastructure based on Open Grid Service Architecture (OGSA) using the BPEL4WS concepts. The OGSA infrastructure (a) exploits a variety of advanced Grid features, such as factories, lifecycle management, and notifications; (b) supports basic workflow functionalities; and (c) uses an orchestration-based approach to Grid service workflow management by specifying the Grid Workflow Execution Language notation and a Grid workflow execution engine. However, the infrastructure is not based on a logical concept. The workflow consists of concrete Grid services that cannot be easily re-used, and no graphical interface is available for constructing workflow models online.

Bubak et al. (2005) introduced a system for constructing Grid workflows consisting of: (i) a workflow composer; (ii) a Grid service registry; (iii) domain ontologies, which must be populated by the user to describe and register data and services; and (iv) an ontology inference engine to acquire the proper data and services. This system is more conceptual than practical. Domain ontologies and semantic descriptions are the key concepts for an automatic workflow system, but without a well-defined, feasible ontology, usage of such a system is limited.

The K-Wf Grid (Babik et al. 2006) uses state-of-the-art semantic Web and Grid technology to facilitate automatic and dynamic orchestration and execution of knowledge-based Grid workflows. The system defines a new XML-based language – the Grid Workflow Description Language – to describe a workflow model that is built based on Petri net theory. It also semiautomatically maps an abstract workflow into a concrete workflow and dynamically monitors and analyzes the workflow performance. The system is similar to a good practical semantic Grid system. However,
geospatial knowledge and standards are not involved, and BPEL4WS is not considered.

Swift (Zhao et al. 2007) is a Grid-based parallel computation environment, which uses XML schema to describe logical structure and its own scripting language to express the workflow. Swift is a task-oriented system for providing super parallel computation in the Grid environment.

Taverna (Oinn et al. 2005) is a Web Services-based tool for composing bioinformatics workflows for the life sciences community, but it provides minimal Grid services.

SwinDeW (Yan et al. 2006) introduces a peer-to-peer (P2P) system to manage decentralized workflows for improving performance, scalability, reliability, user support, and system openness. Dutch Space and the European Space Agency (ESA) European Space Research Institute have proposed a Grid-based workflow management system – GridAssist (Ter Linden et al. 2005) – to provide a user-friendly environment for executing distributed Earth observing instrument simulations using a computational Grid.

Model Builder (ESRI 2008) is a set of commercial software components from ESRI Inc. that provides three types of components for constructing models: elements, connectors, and text labels. Elements are data and pre-implemented tools for users to process data for user-specific applications. Some constructed models can be reused, but each model is application-specific and data-specific. All procedures are manual and instance (specific tools and data)-based. There is no discussion of how to incorporate expert knowledge to logically model abstract geospatial processes using a particular set of service instances and data. Model Builder is a desktop system and not Web-based.

An ontology-based Grid workflow system is a semantic or knowledge system. Zhuge (2004, 2006) introduced a relevant description of such a system, consisting of a methodology, a resource space model, a semantic link network model, and a knowledge flow network model of the semantic Grid and the knowledge Grid.

Compared to previous systems, the research reported here has the following improvements:

(a) Adopts geospatial and information standards, such as geospatial domain standards, Grid standards, and BPEL4WS.

(b) Defines a high-level conceptual model schema using the geo-tree concept for automatically constructing geospatial processing.

(c) Uses a geospatial ontology to define the abstract model.

(d) Provides a user-friendly online graphical user interface for easily capturing experts’ knowledge.

(e) Fully utilizes and integrates the functions of Globus Toolkit 4.0. Grid service is the atom for constructing complicated geospatial applications.

(f) Logical geoprocessing model representing experts’ knowledge can be repeatedly instantiated into concrete workflows by selecting optimal data and Grid service’s instances dispersed in the Grid virtual organization.

(g) A BPEL4WS compatible Grid workflow is designed to execute the concrete workflow as a Grid service-chaining.

3. Theoretical foundation

The VGP uses the geo-object and geo-tree concepts described by Di and McDonald (1999) and subsequently enhanced and enriched (Di et al. 2005, 2006, 2008) at the George Mason University Center for Spatial Information Science and Systems.
A geo-object consists of a data set, the data set attributes (metadata), and a set of methods that operate on the data set (e.g., transformation, subsetting). A geo-object stored at a data center is an archived geo-object. Theoretically, all geoinformation and knowledge products can be derived from archived geo-objects. Thus, from the object-oriented point of view, all processes for discovery of geoinformation and knowledge are procedures of creating new geo-objects from existing ones.

A geospatial logical process model (LPM) is a conceptual model representing the logical relationships among components of a complex geospatial application. A ‘user geo-object’ is a geo-object that results from a user request. This object either is already in a data archive or can be derived by executing a geo-processing algorithm on a set of input geo-objects. Decomposition of a user geo-object produces a tree structure representing the process workflow, called a geo-tree. Construction of a geo-tree is a geospatial modeling process; the geo-tree itself is a geospatial LPM that contains the knowledge of a specific application domain and results in a VGP upon request. Because a geo-tree captures the process steps but not a specific product, it represents only the type of geo-object that it can produce, not an instance (an individual data set). The geo-tree thus logically describes how archived geo-objects work together to generate a new (virtual) geo-object. A geo-tree, with user-specified parameters, can be instantiated from an LPM into a concrete workflow that is executed by the workflow engine to produce the VGP. Within this procedure, the knowledge in the LPM can be easily and widely reused.

Each node of a geo-tree can be considered a geospatial service module. In this module, the service should be well defined, have clear input and output requirements, and be capable of independent execution. This article describes some sample geospatial Grid service modules that can be reused to construct new geospatial models – new geo-trees. From the Grid service point of view, a geo-tree is a complex Grid service chain, and the construction of a geo-tree is a service-chaining process.

4. Geospatial AIM and its XML schema

To support VGP, we present a geo-tree description of an AIM using Service-Oriented Architecture (SOA) and XML representations. The AIM can be used by experts to construct geospatial LPMs and consists of (1) an abstract DataType, (2) a ServiceType, (3) the logical relationships between DataTypes and ServiceTypes, and (4) the cataloged XML descriptions and registration of the above three, to support on-the-fly construction of an LPM.

4.1 Definition of DataType

DataTypes are defined using the scientific terminology of the Earth science disciplines, for example, Digital Elevation Model (dtDEM), land slope (dtSlope), land aspect (dtAspect), and landslide (dtLandslide). A DataType is an abstract type for a data set category. Experts use DataTypes to express their knowledge when developing an LPM using a Web-based interface. The DataTypes and relationships among DataTypes are defined in the geospatial ontologies (Raskin and Pan 2005).

4.2 Definition of ServiceType

A ServiceType represents a set of Grid/Web services that have similar structures and functionalities. The ServiceType Name, Input, Output, and Operation are expressed
using Earth science terminology to reflect the scientific meaning of a service category. The ServiceType Name is the Operation Name, e.g., slope calculation (stSlope) or aspect calculation (stAspect). The Input and Output indicate the DataTypes and ServiceType processes and returns, respectively. The Operation indicates the functions that a ServiceType performs. The relationships between ServiceTypes are defined using a geospatial service ontology (Figure 1).

A ServiceType is specified by its name, attributes, input data type, operation type, and output data type. Table 1 provides the XML definition for ServiceType.

4.3 Definition of logical relationships in AIM

In the AIM, logical activity is introduced to express the logical relationships among ServiceTypes. Four core logical activities are defined: unit-activity, parallel-activity, sequence-activity, and condition-activity (Figure 2.).

Unit-activity has only one member, a ServiceType. Each ServiceType is encapsulated as a unit-activity. The other three activity types are specified by two members, which can be any ServiceTypes or core activities. In Sequence-activity, the first and last members are in sequence, as the output of the first member is the input of the last. In parallel-activity, the action of the two members is in parallel. Condition-activity includes a condition in addition to two members. If the condition is true, the first member acts; otherwise, the last one does.

Table 1. XML definition of ServiceType.

```
<serviceType name="service-type-name">
  <inputList>
    <input name="data-type-name"></input>
  </inputList>
  <operationList>
    <operation name="operation-name"></operation>
  </operationList>
  <outputList>
    <output name="data-type-name"></output>
  </outputList>
</serviceType>
```

Figure 2. Logical relationships in the abstract information model.
Activities with two members are called bi-activities. If both the first and the last members of a bi-activity are unit-activities, the bi-activity is a simple activity. Otherwise, it is called a complex activity. Both simple and complex activities represent workflows and are the final activities exposed to users.

4.4 **AIM specification and its XML schema**

Table 2 presents the formal AIM specification, consisting of a DataType, a ServiceType, and their relationships. Table 3 is the XML schema defining vocabularies and rules used by experts to build up an LPM for a specific application.

4.5 **AIM registration and Grid-enabled geospatial catalog**

A standards-compliant geospatial catalog for registering the AIM, including DataTypes, ServiceTypes, and their relationships, has been designed and implemented. The catalog accepts registration of available data sets and service instances. Subject matter experts retrieve ServiceTypes and DataTypes from the catalog to construct an LPM and register it as a VGP. The registered data sets and service instances are used to automatically instantiate concrete geospatial processing.

Table 2. The XML specification of the AIM.

```xml
<aim>
  <abstractservice name="name" targetnamespace="http://bpel.laits.gmu.edu/aim">
    <service name="sname">
      <attribute input="inname" operation="opname" output="outname"/>
    </attribute>
    <activity>
      <activitymember name="mname" operation="opname"/>
    </activitymember>
    <compositeactivity>
      <unitactivity name="uaname" member="mname"/>
      <sequenceactivity name="saname" fmember="mname" lmember="mname"/>
    </compositeactivity>
  </abstractservice>
</aim>
```

Table 3. Vocabularies and rules for the AIM.

```xml
<x:element name="service" type="as:serviceType">
  <x:unique name="attribute"/>
  <x:selector xpath="attribute"/>
  <x:field "@name"/>
  <x:complexType name="serviceType">
    <x:complexContent>
      <x:sequence>
        <x:element ref="as:attribute" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
    </x:complexContent>
  </x:complexType>
</x:element>
```

---

*Modeling virtual geospatial products* 587

Downloaded by [Tsinghua University] at 18:41 10 June 2012
workflows from LPMs when users request a VGP. A Web interface of the catalog is provided for users querying both real and virtual data products.

An object-oriented, Grid-enabled Catalog Service for the Web (GCS/W) (hereinafter referred to as 'the catalog') has been designed. The catalog follows the e-business Registry Information Model (ebRIM)-based (OASIS 2002) and the OGC Catalog Service for the Web (CS/W) specification (Gilles 2006, Primavera 2006) for registering basic objects and the AIM. It extends the CS/W information model for registering geospatial data sets and service instances. This model is based on the NASA Earth Observing System (EOS) Core System metadata standard for HDF-EOS data (NASA 1994), the FGDC Content Standard for Digital Geospatial Metadata and its Remote Sensing Extensions (FGDC 2002), the ISO 19115 standard Geographic information – metadata, and ISO 19119 Geographic information – services. Details can be found in Section 6.2. The catalog conforms to the OGC CS/W ISO profile (Voges et al. 2005).

EbRIM provides features for users to define classification schemes, and the ClassificationScheme object is used to describe taxonomies. The DataType classifications are defined as in the NASA Global Change Master Directory (GCMD). ServiceTypes are defined in three ways, according to the NASA GCMD, ISO 19119, and OGC service standards. There is thus one DataType classification and three ServiceType classifications. Each DataType or ServiceType corresponds to a ClassificationNode object in the GCS/W. Every DataType and ServiceType is registered in the GCS/W. One kind of DataType corresponds to only a single DataType classification, but a ServiceType may belong to one or more ServiceType classifications. The AIM is registered in GCS/W and is later used by experts to build up LPMs. The GCS/W plays a key role in modeling virtual products, because DataTypes, ServiceTypes, LPMs, data sets, service instances, and their related metadata are all stored in and accessible from the catalog.

5. Modeling the logical processing model

Using the AIM, a Web-based, user-friendly interface is provided for experts who are logically and virtually constructing an LPM. An expert constructs an LPM to virtually declare a geospatial product that does not exist in a geospatial system, but can be created upon user request. The LPM provides product virtualization, enabling a geospatial data system to offer an unlimited number of products with little effort. The LPM is also a kind of special ServiceType that can be repeatedly used by domain experts to widely share and reuse knowledge.

Modeling an LPM consists of the following three steps:

(a) Experts launch a User Web Interface (UWI) that is designed and implemented using the AIM.
(b) Experts construct abstract LPMs using the available DataTypes and ServiceTypes.
(c) Experts register LPMs into the catalog as VGPs for future retrieval. LPMs are, in fact, new ServiceTypes that can be reused to construct other LPMs.

5.1 Design of the user web interface

The purpose of the UWI is to provide a user-friendly Web-based interface for experts to express their domain knowledge – creating LPMs based on the AIM. Table 4 lists the requirements that guide the design of the UWI.
The UWI runs as a Java Applet in Web browsers. The interface is divided into two columns (Figure 3). The left column contains three tab pages: DataType, ServiceType, and Selected ServiceType. The DataType and ServiceType tab pages list all available resources as tree structures. The Selected ServiceType tab page lists available resources to experts while querying the catalog.

The right column contains two tab pages: Model Design Panel and Registered Model List. The Model Design Panel is a workbench on which experts construct LPMs. On the workbench, a DataType is represented as a green octagon and a ServiceType is represented as a blue rectangle. A ServiceType rectangle is connected with one or several DataType octagons as input and one DataType octagon as output. Pairs of

<table>
<thead>
<tr>
<th>Table 4. The requirements for guiding the design of the UWI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a  The UWI should automatically display all available DataTypes and ServiceTypes in the system catalog for experts to select</td>
</tr>
<tr>
<td>b  Experts should be able to query and display all available ServiceTypes while selecting an existing DataType, and vice versa</td>
</tr>
<tr>
<td>c  Simple descriptions should be provided for every available DataType and ServiceType</td>
</tr>
<tr>
<td>d  There are no direct connections between any two DataTypes or ServiceTypes. There must be a ServiceType between two DataTypes, and vice versa</td>
</tr>
<tr>
<td>e  LPMs should be WYSIWYG (What you see is what you get)</td>
</tr>
<tr>
<td>f  The UWI should provide a mechanism for validating the final LPM</td>
</tr>
<tr>
<td>g  The UWI should provide detailed procedures for input of metadata during registration of the LPM to the catalog</td>
</tr>
<tr>
<td>h  Experts and general users should be able to list all registered LPMs and authorized experts should be able to modify LPMs</td>
</tr>
</tbody>
</table>

The UWI runs as a Java Applet in Web browsers. The interface is divided into two columns (Figure 3). The left column contains three tab pages: DataType, ServiceType, and Selected ServiceType. The DataType and ServiceType tab pages list all available resources as tree structures. The Selected ServiceType tab page lists available resources to experts while querying the catalog.

The right column contains two tab pages: Model Design Panel and Registered Model List. The Model Design Panel is a workbench on which experts construct LPMs. On the workbench, a DataType is represented as a green octagon and a ServiceType is represented as a blue rectangle. A ServiceType rectangle is connected with one or several DataType octagons as input and one DataType octagon as output. Pairs of

![Diagram of LPM](image-url)

Figure 3. The user Web interface for building logical geospatial process models.
ServiceTypes can be connected only through a DataType that is the output of the first ServiceType and the input of the second. The Registered Model List panel is used to provide user-friendly management of all registered LPMs. Authorized experts can list all registered LPMs, view the details of the descriptions, reload them on the workbench for modification, or delete them. Figure 3 displays a screenshot of a landslide susceptibility example having four inputs.

5.2 Expressing the knowledge of experts

An expert uses an LPM to represent the steps in processing raw data into an application-specific geospatial product. When an expert opens the UWI to start building a LPM, the UWI automatically displays the available DataTypes and ServiceTypes from the catalog. Now, the expert has to decide what ‘type’ of data he wants – the ‘type’ is one of the DataTypes defined in the AIM. The expert selects one desired DataType. He selects a ServiceType that, he thinks, can produce that type of data. He then selects one or more new ServiceTypes whose output (DataTypes) matches the input (DataTypes) of the previous ServiceType selected (refer to Figure 3). Repeating the last step, the expert each time uses the output of the new ServiceType as the input of the ServiceType selected in the previous step until the input for the last ServiceType is an elementary DataType whose corresponding DataInstances already exist. This process creates a ServiceType chain that is an LPM, representing the domain knowledge on how to produce a specific geospatial product. Using the AIM, an expert can build LPMs from any combination of sequence, parallel, and condition activities.

To avoid producing invalid LPMs, the constraints discussed in Sections 4.1 and 4.2 are applied when DataTypes and ServiceTypes are defined. An expert can query all ServiceTypes from the catalog consistent with these constraints after selecting a DataType. The query results will be shown in the left column of the Selected ServiceType tab page, and the expert can select only these ServiceTypes. The procedure is similar when the expert first selects a ServiceType and then needs to select a DataType. Before an LPM is registered into the catalog, each ServiceType is queried to verify that the inputs and output DataTypes are consistent with the ServiceType requirements.

5.3 Registration and retrieval of LPMs

An expert can register a new LPM into the catalog as a VGP. The logical model created in the workbench is converted into an XML document following the XML schema of the AIM. This XML document is registered in the catalog.

The queriable core metadata elements, a subset of the extended CS/W information model described in Section 4.5, are applied to describe the LPM. These elements are registered into the catalog with the LPM. Thus, the expert can describe the new LPM using standard FGDC or ISO 19115 metadata: name, description, applicable spatial and temporal range, general keywords, specific keywords, etc. Figure 4 shows screenshots of an LPM registration.

An LPM that has been built and registered in the catalog is immediately available for user queries. A universal CS/W portal interface was designed and implemented for querying both the real data products and the LPMs using the same queriable core metadata elements. The user never needs to know if the desired product already exists or must be produced online. However, there are significant differences in the internal
procedures of the two types of requests. When replying to a VGP request, the system receives only a qualified LPM that represents a VGP that does not really exist. The LPM is instantiated into a concrete processing workflow consisting of Grid services and executed through a Grid workflow engine to produce the requested VGP.

5.4 An example of an LPM

Figure 3 shows an LPM for Landslide susceptibility forecasting with four inputs: Terrain_Slope, Terrain_Aspect, California_WHRI3_Class, and Enhanced Thematic Mapper (ETM)_Normalized Difference Vegetative Index (NDVI) DataTypes, which are the outputs of the Slope, Aspect, California_WHRI3_Classification, and NDVI ServiceTypes, respectively. The input DataTypes of the Slope and Aspect ServiceTypes are of DataType Terrain_Elevation, and the inputs of California_WHRI3_Classification and NDVI ServiceType are of DataType – ETM. Table 5 summarizes the LPM as registered in the catalog.

6. Instantiating LPMs into grid service-oriented concrete workflows

When a user requests a virtual product, the system will materialize the LPM using those concrete data sets and service instances with attributes available in the Grid environment. Therefore, data sets, service instances, and their metadata must be available in the catalog. A Grid INstantiation Service (GINS) was created to instantiate LPMs into executable concrete workflows. GINS involves the following key steps:
(a) Define service instances, data sets, their metadata descriptions and their registration to the catalog;
(b) Parse the LPMs;
(c) Select appropriate service instances and data;
(d) Transfer parameters of service instances along sequences of service instances; and
(e) Produce Web Service Description Language (WSDL) and BPEL descriptions of the concrete workflow.

6.1 Definitions of DataInstance and ServiceInstance

DataInstances (also referred to as data sets) are instances of some kinds of DataType and physically exist. Data sets may have multiple replicas resulting from multiple requests for a virtual product across distributed storage location in the Grid environment. These data sets will be accessed when a user requests both existing data and VGPs, but will never be accessed in the modeling phase when an expert is constructing a VGP.

A ServiceInstance is a service instance that belongs to one or several ServiceTypes. It is physically deployed in many kinds of distributed computing units in the Grid environment, such as high-performance servers or clusters. All ServiceInstances are standard Grid services that are described through WSDL files, associated with one or several ServiceTypes, and used upon request to construct concrete workflows.

Figure 5 shows the relationships among ServiceTypes, DataTypes, ServiceInstances, DataInstances, and LPMs. The output of the LPM is a previously

![Image of Figure 5: Relationships of DataType, ServiceType, DataInstance, and ServiceInstance.](image-url)
existing DataType. DataInstances are associated with one or more DataTypes, whereas ServiceInstances are associated with one or more ServiceTypes. DataInstances refer to both real data and virtual data created upon request.

6.2 Metadata descriptions of DataInstance or ServiceInstance and their registration

The core workflow metadata are described using the ISO 19115 metadata standard parts 1 and 2 (ISO/TC211 2002a, 2002b), the NASA ECS metadata standard (NASA 1994) for HDF-EOS data, and the FGDC metadata standard and its extension for remote sensing (FGDC 2002, Neal et al. 2006). Chen et al. (2005) list the core metadata elements the system selects for a data set. This extended metadata model is integrated with the OGC CS/W information model. A CSWExtrinsicObject object is used to represent a DataInstance. The Slot object is used to provide metadata elements that are not original attributes of CSWExtrinsicObject.

The metadata elements are represented in an XML document following the ISO 19115 metadata XML schema. A data set is registered in the catalog by processing its metadata document. If a data set has multiple replicas on the Grid, a unique logical name is defined. Each replica is associated with a concrete service instance that has an accessible physical address for serving this replica upon user request. The Replica Location Service (RLS) (Chervenak et al. 2008) distributed as part of Globus Toolkit is used to store the relationships between logical names and concrete replicas. Registration of a data set involves an ebRIM-based catalog service and Globus RLS, both of which must be registered with related metadata.

The ISO 19119 service metadata standard (ISO/TC211 2002c) is used to describe service instances. The Slot object is used to extend service object elements when service instances are registered to the catalog. Each service instance has a standard WSDL description. To chain a service instance correctly as a part of the entire concrete workflow, the Instantiation service must parse the WSDL to get the input and output parameters of the selected service instance. Table 6 summarizes a concise WSDL example of the Grid Slope service.

6.3 Parsing of LPMs and optimized selection of service instances and data sets

The GINS includes a pure Java parser, which parses an LPM to extract the ServiceType chains from the XML document. The parser lists all ServiceTypes involved in the LPM and their logical relationships.

Usually, several instances of a given ServiceType and several replicas of a given data set are available in the system. A Replica and Optimization Service (ROS) was created to automatically select the optimal service instance and data set replicas. The ROS integrates the Monitoring and Discovery Service (MDS4) and the RLS of Globus Toolkit 4. The RLS is tightly coupled with GCS/W to provide data replica management of registered geospatial data and selection of optimal service instances and data sets. MDS4 monitors and obtains dynamic information for every node. RLSs are responsible for managing distributed replicas of registered geospatial data. An ROS identifies all nodes that provide service instances or/and data sets, then selects that node with optimal available resources such as CPU, memory, and storage.

After GINS parses the LPM to get all ServiceTypes, it queries the catalog by submitting each ServiceType as a query condition to GCS/W, to retrieve all available service instances for each individual ServiceType. Then, ROS is invoked to select the service instance that provides the best performance in the Grid environment for each
ServiceType. The list of ServiceTypes becomes a list of optimal service instances. The WSDL file of each service instance is also returned. Given the appropriate WSDL file and the list of logical relationships, the mapping of parameters between any sequentially connected service instances can be determined. An XML document expressing this mapping is used to produce the BPEL file of the LPM.

The Grid-enabled Web Coverage Service (GWCS) and Grid-enabled Web Map Service (GWMS) (Chen et al. 2006a) are always used to process raw geospatial data to provide data products or intermediate results that meet user query requirements. Every raw geospatial data item is bound to a GWCS and/or a GWMS. The GWCS/GWMS is always automatically chained into the concrete workflow to serve raw geospatial data to those individual service instances that require the data as an input. Therefore, the selection of the optimal raw data is related to the selection of the GWCS/GWMS. Because several replicas of the raw data exist in the Grid environment, it is best to select the raw data at a node where a GWCS/GWMS exists. If several nodes have both data and GWCS/GWMS, then the ROS is invoked to select the node with the best performance. If GWCS/GWMS is not available at the node with the desired raw data, the best node that hosts GWCS/GWMS and the best node that hosts the desired raw data are selected. Then, GridFTP-based DTS securely transfers the desired raw data to the best GWCS node, where the raw data are processed.

### 6.4 Producing WSDL and BPEL files for concrete workflow

Given the list of available optimal service instances, their WSDL files, and the list of logical relationships between service instances, GINS produces the BPEL (Curbera et al., 2006) and WSDL descriptions necessary to execute the concrete workflow. Tables 7 and 8 provide partial examples of a concrete BPEL workflow and a WSDL.
**Table 7.** Part of a concrete BPEL workflow file.

```xml
<BPELProcess xmlns="http://bpel.laits.gmu.edu/bpel/">
  <process ... name="Landslide_2i" targetNamespace="http://bpel.laits.gmu.edu/bpel/lpm">
    <partnerLinks>
      <partnerLink name="Landslide_2iProvider"
        partnerLinkType="tns:Landslide_2iPLinkType"/>
      <partnerLink name="GridWCSServiceProvider" .../>
      <partnerLink name="GridSlopeAspectServiceProvider" .../>
      <partnerLink name="GridSlopeServiceProvider" .../>
    </partnerLinks>
    <variables>
      <variable name="request" messageType="tns:Landslide_2iRequestMessage"/>
      <variable name="response" messageType="tns:Landslide_2iResponseMessage"/>
    </variables>
    <sequence name="main">
      <receive name="receive"
        partnerLink="Landslide_2iProvider"
        portType="tns:Landslide_2iPortType"
        operation="process">
        <flow>
          <sequence>
            <assign>
              <copy>
                <from expression="string('[564809.0,4161608.0,587849.0,4192328.0]')"/>
                <to variable="getCoverageInputMessage_3Request" part="parameters" query=""/>
              </copy>
            </assign>
          </sequence>
        </flow>
      </receive>
      <invoke partnerLink="GridWCSServiceProvider"
        portType="ns0:GridWCSPortType"
        operation="getCoverage" inputVariable="getCov_3Request" outputVariable="getCov_3Response">
        <reply name="reply" partnerLink="Landslide_2iProvider" portType="tns:Landslide_2iPortType">
          <copy>
            <from expression="string('epsg:32610')"/>
            <copy>
              <from expression="string('application/HDF-EOS')"/>
            </copy>
          </copy>
        </reply>
      </invoke>
    </sequence>
  </process>
</BPELProcess>
```

**Table 8.** Part of a WSDL file for a concrete BPEL workflow file.

```xml
<wsdl:definitions name="Landslide_2i" targetNamespace="http://bpel.laits.gmu.edu/bpel/lpm">
  <wsdl:import WSDL files for GridWCS, GridLandslide2, GridSlopeAspect, and GridSlope />
  <wsdl:types>
    <s:element name="Landslide_2iRequest" .../>
    <s:element name="Landslide_2iResponse" .../>
  </wsdl:types>
  <wsdl:message name="Landslide_2iResponseMessage" .../>
  <wsdl:message name="Landslide_2iRequestMessage" .../>
  <wsdl:portType name="Landslide_2iPortType">
    <wsdl:operation name="process">
      <wsdl:input message="tns:Landslide_2iRequestMessage"/>
      <wsdl:output message="tns:Landslide_2iResponseMessage"/>
    </wsdl:operation>
    <plnk:partnerLinkType name="GridWCSServicePLinkType"/>
    <plnk:partnerLinkType name="ns1:GridWCSPortType"/>
    <plnk:partnerLinkType name="GridSlopeAspectServicePLinkType"/>
    <plnk:partnerLinkType name="GridSlopeServicePLinkType"/>
    <plnk:partnerLinkType name="GridLandslide2ServicePLinkType"/>
  </wsdl:portType>
</wsdl:definitions>
```
file, respectively, for the landslide susceptibility application, with Slope and aspect as two inputs.

7. Grid workflow engine and execution of BPEL workflow

BPEL is being used by OGSA and the Web Services Resource Framework to implement a Grid service workflow (Slomiski 2006). Recently, many Grid workflow engines have been created, for example, PEGASUS (Deelman et al. 2006), K-WfGrid (Bubak et al. 2006), Swift (Zhao et al. 2007), Gridbus (Yu and Buyya 2006), Kepler (Ludascher et al. 2005), Taverna (Oinn et al. 2005, Hull et al. 2006), and SwinDeW-G (Yang et al. 2007). A BPEL-compliant Grid Workflow Engine Service (GWES), named GridBPELPower, has been designed for executing a concrete BPEL workflow. Users can get their desired VGPs by using GWES to execute the concrete BPEL workflow. GridBPELPower consists of the following functional modules:

1. Grid WSDL services that provide online operation for deploying, initiating, testing and running Grid services, and viewing the WSDL of Grid services (see Figure 6). A Grid service can be deployed by providing its local or online WSDL file in secure or insecure mode.

2. A BPEL process that can be deployed using a zip/jar file or two files: *.bpel and *.wsdl. The file *.bpel includes the concrete workflow represented in BPEL, and *.wsdl is the WSDL description of the concrete workflow.

![Figure 6. GridBPELPower interface, showing some deployed grid services and one service's initiating interface.](image-url)
(3) Instances that provide tools for instances of BPEL processes: testing, debugging, and viewing the source.

(4) Logical Business Process Library (LBPL) activities that provide tools for viewing LPMs and for debugging each step of a model. A logical model is deployed through its URL address.

GridBPELPower works in two modes. The first mode is to execute a Grid services workflow. GridBPELPower can be deployed into any Grid service container, such as the Globus container, and securely invoked by any Grid service using the BPEL and WSDL files. The second mode is as a Grid services container. Grid services can be deployed into GridBPELPower through their WSDLs. Users can also deploy concrete BPEL workflows through the BPEL and WSDL files, and can test, debug, and run workflows, view workflow charts, and view and debug WSDL files of the Grid services involved. GridBPELPower serves as a Grid services orchestration and choreography engine in the SOA environment, integrating Grid services into collaborative and transactional geospatial processes. It supports different invocations of Grid services, such as the HTTP POST/GET and SOAP document/rpc (remote procedure call) styles. Both orchestrated Grid services and Web services can be executed using this engine. It can be run in J2EE-compliant application servers, for example, Tomcat, JBoss, Sun AS, Oracle AS, Weblogic, and WebSphere. Figure 6 shows the interface of GridBPELPower as a Grid service container, with some deployed Grid services and the initial procedure of one of the Grid services.

8. System architecture, implementation, performance, and application

8.1 System architecture

The system architecture consists of three tiers (Figure 7). The first tier includes a modeling interface for experts to construct LPMs, and data retrieval and acquisition interfaces for both experts and general users. The second tier, the OGC Web Services interface tier, consists of OGC CS/W and WCS/WMS portals. With these portals, users can transparently access Grid computing resources and Grid-based geospatial resources. The third tier, the Grid services tier, includes Grid-enabled OGC Web services, extended Grid services, and Grid geospatial services (Chen et al. 2006b).

Subject matter experts construct abstract processing models using the modeling interface and register models in the catalog using the GCS/W interface. General users deploy the CS/W portal to submit requests for geospatial products. A list of matched geospatial products, both real and virtual, is returned to the user interface. User selections from the list are sent to the Grid-enabled Virtual WCS/WMS (GVWCS/WMS) service, which forward the requests to the Intelligent Grid Service Mediator (iGSM) for real data and to the GINS for virtual products. GINS parses the LPMs to extract the ServiceTypes, then queries GCS/W and invokes ROS to obtain the optimal service instances and their WSDL descriptions for every ServiceType. Finally, GINS produces BPEL and WSDL files and returns them to GVWCS/WMS. GVWCS/WMS sends both files to GWES – GridBPELPower. GWES executes the BPEL workflow by invoking Grid services instances distributed in the Grid environment to generate products and returns the URL from which users can get the products.
8.2 Testbed and system security

The testbed is based on OGSA (Foster et al. 2002). The Virtual Organization (VO) forms a secure Grid environment for the testbed. The testbed VO consists of the GMU CSSI, NASA ARC, and LLNL Earth System Grid (ESG). Each partner’s VO has its own Certificate Authority (CA), issuing host, user, and services certificates for authorized users. Authentication and authorization between any two VOs and any two machines in the VO are possible.

Because this Grid project serves as one node of the Committee on Earth Observation Satellites (CEOS) Grid testbed, any two machines from the CEOS VO are joined to the testbed VO. Authorized users from the CEOS testbed can access any Grid resource in the testbed, and users of the testbed can use any resource located at the CEOS node.
8.3 The lifecycle of VGPs

The lifecycle of Grid-enabled virtual geospatial process modeling consists of three phases:

1. The knowledge phase – building LPMs by composing geospatial modules into geospatial processes. The experts’ knowledge is embedded into individual modules and the logical relationships of composed modules.
2. The instantiation phase – instantiating LPMs into concrete Grid service chains from available optimal Grid services instances and data listed in the system catalog.
3. The product phase – executing the concrete workflow, geospatial Grid service chains to generate geospatial products for users.

When users request virtual data, the second and third phases form a new short lifecycle to serve VGPs. The users thus benefit from the professionals’ knowledge.

8.4 Performance evaluation

The dependence of overall request-response times of a Grid service (secure and non-secure) and a Web service on the method of invocation, the size of the request, and the response payload have been compared. One Grid service (GCS/W)-Web service (CS/W) pair was compared as a function of request and response payload sizes. GCS/W runs in the Globus container and CS/W runs in an Axis Web service engine deployed in the Tomcat container. Both services use the same core Java catalog services, database management system, geospatial data catalog, access protocols, and running environments.

The secure Grid service requires 300 milliseconds more processing time than the non-secure Grid service and 600 milliseconds more processing time than the Web service. As payloads increase, these differences in response times converge to a small number as does the ratio of the overhead to the overall processing times, and eventually can be ignored. Those performances of the Web service and Grid service are tested using a single PC. The experiment shows that both kinds of single services have almost identical performances with increasing payload while they run alone. However, faced with a global complex geospatial application, the advantages of Grid services, such as Grid security, finding and selecting optimal data and computing resources from available huge amount of online resources, stand out. However, our utilization of grid computing did not focus on the improvement of the system performance by optimizing the collaboration among components. A systematic investigation of the performance bottlenecks and their solutions within a Web platform (such as Yang et al. 2005) and geospatial Web services (such as Yang et al. 2007) would be needed for future improvement of the system.

8.5 Application case and experiment results

As an example, landslide susceptibility will be generated for regions of California where there are DEM data in HDF-EOS format. Professionals want such geospatial data products. How can experts utilize the system to reach this goal?

An expert can easily derive landslide susceptibility from the slope and aspect ratio of the terrain, parameters that can be obtained from the DEM data through the appropriate Grid services. Using the modeling graphic interface, the expert selects the Landslide_Susceptibility DataType. An instance of this DataType is the expert
desired data. By querying the system catalog, the expert knows that the Landslide_Susceptibility DataType can be produced by either of two ServiceTypes: the Landslide_Susceptibility_2i ServiceType, which requires two inputs, and the Landslide_Susceptibility_4i ServiceType, which requires four inputs. The expert selects the Landslide_Susceptibility_2i ServiceType, which has the required input DataTypes: Terrain_Slope and Terrain_Aspect. The slope and aspect ServiceTypes can produce Terrain_Slope and Terrain_Aspect DataTypes, respectively, given the DEM DataType. The DEM data for the terrain are available in the system in HDF-EOS format and can be directly served through GWCS. The expert creates the LPM and registers it in the system catalog for future reuse. Figure 8 shows the virtual data product produced – the landslide susceptibility of a region in California.

9. Conclusions and future research

This article has introduced an intelligent Grid-enabled approach to implement VGPs. This approach facilitates domain experts’ extensive use of Earth science knowledge to produce VGPs for research and applications. An AIM has been proposed to conceptualize and classify geospatial domain ontologies for building up LPMs. A Web interface was created to facilitate the logical composition of geospatial process models. All LPMs that are constructed can be registered into a Grid-enabled system catalog. A BPEL-compliant instantiation Grid service was developed to instantiate LPMs into Grid service-oriented concrete workflows, as Grid service chains. Finally, a BPEL-compliant Grid workflow engine was developed to execute concrete workflows in Grid environment to produce virtual data products. This approach utilizes
the three phases of knowledge, instantiation (by optimizing available information), and product (in the form of data). Ultimately, a user can apply knowledge derived from experts with Grid resources to efficiently and rapidly obtain valuable geospatial products. This new system will facilitate the use of Grid technologies and OGC Web services for resource sharing and interoperability in geospatial research.

A promising development of an ontology-based collaborative semantic and knowledge Grid (Flahive et al. 2005) has been inspired by intelligent knowledge systems (Di 2005, Oh et al. 2006) based on Web services. Existing geospatial taxonomies can be used to develop the geospatial domain ontologies (Raskin and Pan 2005, Di et al. 2006). An automatic geospatial inference system will be implemented in the future for automatic intelligent integration of Grid services, supported by Grid resources and combined with our current system, geospatial ontologies, and Web service integration tools. LPMs with knowledge incorporated will be constructed to produce many VGPs. Visualization and monitoring of execution procedures and behavior of workflow chains over the Grid is important (Allcock et al. 2002). Doing so will help find failure locations, bottlenecks, and other problems.

Acknowledgments
This project was supported by grants from the NASA Earth Science Data and Information System Project (NCC5-645, PI: Dr. Liping Di), NASA Advanced Information System Technology program (NAG-13409, PI: Dr. Liping Di), NASA REASoN program (NNG04GE61A, PI: Dr. Liping Di), and US National Geospatial-Intelligence Agency NURI program (HM1582-04-1-2021, PI: Dr. Liping Di). Additional funding was supplied by OGC for developing the OGC Web Services: WCS, WMS, CS/W. Many thanks are given to our colleague, Dr. Barry Schlesinger, for his proofreading of the manuscript.

References


YANG, W. and WHITESIDE, A., 2005, Web image classification service (WICS) implementation specification (OGC®-05-017), version 0.3.3. Open Geospatial Consortium Inc., 8 February, pp. 15–32.


