Prototyping an online wetland ecosystem services model using open model sharing standards

Min Feng\textsuperscript{a,b,}, Shuguang Liuc, Ned H. Euliss Jr.\textsuperscript{d}, Claudia Young\textsuperscript{e}, David M. Mushet\textsuperscript{d}

\textsuperscript{a}State Key Laboratory of Resources and Environment Information System, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China
\textsuperscript{b}University of Maryland, College Park, MD 20742, USA
\textsuperscript{c}U.S. Geological Survey (USGS), Earth Resources Observation and Science (EROS) Center, Sioux Falls, SD 57198, USA
\textsuperscript{d}USGS Northern Prairie Wildlife Research Center, 8711 37th St SE, Jamestown, ND 58401, USA
\textsuperscript{e}ADNET Systems Inc., Contractor to USGS EROS Center, Sioux Falls, SD 57198, USA

Abstract

Great interest currently exists for developing ecosystem models to forecast how ecosystem services may change under alternative land use and climate futures. Ecosystem services are diverse and include supporting services or functions (e.g., primary production, nutrient cycling), provisioning services (e.g., wildlife, groundwater), regulating services (e.g., water purification, floodwater retention), and even cultural services (e.g., ecotourism, cultural heritage). Hence, the knowledge base necessary to quantify ecosystem services is broad and derived from many diverse scientific disciplines. Building the required interdisciplinary models is especially challenging as modelers from different locations and times may develop the disciplinary models needed for ecosystem simulations, and these models must be identified and made accessible to the interdisciplinary simulation. Additional difficulties include inconsistent data structures, formats, and metadata required by geospatial models as well as limitations on computing, storage, and connectivity. Traditional standalone and closed network systems cannot fully support sharing and integrating interdisciplinary geospatial models from variant sources. To address this need, we developed an approach to openly share and access geospatial computational models using distributed Geographic Information System (GIS) techniques and open geospatial standards. We included a means to share computational models compliant with Open Geospatial Consortium (OGC) Web Processing Services (WPS) standard to ensure modelers have an efficient and simplified means to publish new models. To demonstrate our approach, we developed five disciplinary models that can be integrated and shared to simulate a few of the ecosystem services (e.g., water storage, waterfowl breeding) that are provided by wetlands in the Prairie Pothole Region (PPR) of North America.

Software availability

Name of software: Wetland Ecosystem Services Model Prototype
Developers: Min Feng, Shuguang Liu
Contact information: feng.tank@gmail.com
Hardware required: None
Software required: Internet browser (Firefox, Internet Explorer, etc.)
Program language: J2EE (server) and client script technologies (JavaScript, XML, etc.)

Source code: Available at http://code.google.com/p/geoengine/ with “Apache License 2.0” code license.
Availability and cost: Users can access the ecosystem services simulation Web site directly at http://wetland.geoclouds.info:59080/wetland/view.do and the model service Web site can be accessed at http://wetland.geoclouds.info:59080/rest/wps for the model service. The prototype system doesn’t require user registration.

1. Introduction

Great interest currently exists in forecasting how ecosystem services may change under alternative land use and climate futures. Ecosystem services are diverse and include supporting services or functions (e.g., primary production, nutrient cycling), provisioning
services (e.g., wildlife, groundwater), regulating services (e.g., water purification, floodwater retention), and even cultural services (e.g., ecotourism, cultural heritage) (Ruth and Stefano, 2005). Hence, knowledge of numerous scientific disciplines is required to successfully develop models that can forecast multiple and simultaneous ecosystem services. Forecasting simultaneous change in multiple ecosystem services will be especially challenging, even for small ecosystems, because traditional forecasting was conducted by scientists within individual disciplines where a consistent means has rarely been used to develop the models, algorithms, and even the data structure among different scientific disciplines (Denzer, 2005). In addition, ecosystem disciplinary models often involve exchanging and analyzing geospatial data, which makes interoperability more difficult (Rao et al., 2007; Lee and Percival, 2008). These differences must be resolved before it is possible to make disciplinary models interoperable for examining interdisciplinatory ecological relationships and the dynamic evolution of ecosystems (Rizzoli and Young, 1997; Oxley et al., 2004).

It is unlikely that traditional standalone and closed network systems (systems that either running on an intranet or using closed communication protocols, such as commercial private standards) will provide the storage or support for computing large geospatial models to forecast change in diverse ecosystem services. Numerous scientific computational models have been developed (Goodchild, 2005), but little progress has been made in model sharing (Liu et al., 2002). Most scientific computational models are command-line applications written in FORTRAN, C, or other programming languages. These models are usually platform-dependent and inaccessible though a network. For applications that require multiple disciplinary models from many modelers, these models need to be collected and set up on a single computing environment, such as a PC or a supercomputer. Besides the difficulties of setting up the computing environment for diverse models, it is usually very challenging to integrate these models because (1) semantic structure and format of model input/output data are heterogeneous, and (2) model coupling procedures do not exist or are not consistent. Keeping models in a centralized location also increases the difficulty for model users since it adds the additional burdens of tracking model versions and keeping them updated (Rao et al., 2007). Some disciplinary models are also location-specific, which may render them unusable elsewhere; thus, model metadata is required for model users to understand and use a particular model (Kouzes et al., 2009).

Some disciplinary models have been integrated into complex decision support systems (DSS) for consideration of environmental and ecological aspects for the sustainable management of land and water ecosystems (Matthes et al., 2007; Rizzoli and Young, 1997; Oxley et al., 2004; Denzer, 2005; Rao et al., 2007). However, the model components within the systems were generally designed to run on standalone computers or in closed networks, rather than sharing and integrating resources in support of interdisciplinary model simulations (Goodchild, 2005). Since the end of last century, systems have been designed to reuse geospatial models (Leavesley et al., 1996; Morozov et al., 2006; Granell et al., 2007). However, the systems were built using inconsistent data formats and model coupling methods. In addition, some model modules and source codes are copyrighted and require special legal agreements for use. In recent years, communities have developed modeling frameworks (e.g., the Community Surface Dynamic Modeling System (CSDMS, http://csdms.colorado.edu), the Earth System Modeling Framework (ESMF: http://www.earthsystemmodeling.org/)) to facilitate model programming and model integration in certain research fields (Voinov et al., 2008). However, little emphasis has been given on interoperability with models from other modeling systems, especially model sharing and integrating over the Internet. The OpenMI (http://www.openmi.org/) proposed a standardized way of linking models from different modeling frameworks (Gregersen et al., 2007), but the OpenMI still does not support linking models through a network.

Geospatial theories and technologies have improved over the last thirty years, and Geographic Information System (GIS) has evolved from desktop to a network based single-tier, multi-tier, and service-oriented architecture (SOA) (David, 2005). Compared to other systems, the SOA-based system has the advantage of sharing and reusing resources over wide geographic regions and different systems (Tsou and Buttenfield, 2003). Sharing geospatial models based on the SOA provides at least two advantages. First, it provides the infrastructure required to integrate different data and models from anywhere in the world to address sophisticated applications (Diaz et al., 2008). Second, SOA-based model sharing is highly amenable to various applications, including Browser/Server and Client/Server, and distributed computing architectures, such as GRID computing and cloud computing (Grimshaw et al., 2009).

The Open Geospatial Consortium (OGC) and the International Standards Organization (ISO) published standards for geospatial data services, including the Web Map Service (WMS), the Web Feature Service (WFS), and the Web Coverage Service (WCS). Open Geospatial Consortium (2007) also published the first version of the Web Processing Service (WPS) standard, which defines a standardized interface to facilitate publication, discovery, and consumption of those geospatial processes by users through a network. Standards from the OGC and ISO (e.g., geospatial metadata) can solve interoperability problems in distributed ecosystem models where sharing and integration are required. The standards have made it possible to openly share geospatial disciplinary models over the Internet to address the needs for simulating diverse ecosystem services using data and model sharing paradigms.

In this paper, we propose an architecture for sharing and integrating ecosystem models, which details how to construct a model service interface and a software platform to help modelers publish their models. We demonstrate an application of the sharing approach we developed using five scientific computational models to simulate some wetland ecosystem services in the Prairie Pothole Region (PPR) of North America. We choose to model wetlands because the ecosystem services they provide span diverse scientific disciplines (Boyd and Banzhaf, 2007), which pose substantial interoperability and integration difficulties to simulation modelers (Cockerill et al., 2006).

2. Model sharing infrastructure

2.1. Architecture of services

The fundamental idea behind SOA is to implement software applications as services to allow clients to use them through the Internet. The concept is based on defined standards, including communication standards (e.g., Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP)) and interoperability standards (e.g., OGC standards). The standards work together to facilitate interoperability among heterogeneous systems (e.g., system with different hardware or software), whether within an individual organization or across the Internet (Wahib et al., 2008). In order to facilitate sharing and accessing geospatial models for integrated ecosystem simulation, we outline an SOA-based open architecture in Fig. 1, which provides a general overview of the key components and their interactions through the Internet. Each component represents either a service that provides callable functions or a client that calls the functions from services. When a client establishes conversation with a server, the client and server
will follow mutually understood standards (interoperability elements) regardless of differences in their software or hardware implementation. Besides providing channels for model interaction, sharing and accessing a model also involves intensive data exchange (also presented in Fig. 1).

The key components in the architecture are introduced below:

1) Geospatial model services

Instead of providing model module copies, the geospatial model services share model computing capability through the Internet as SOA services, and they allow model clients to call the models hosted on the services remotely. The services accept SOAP and HTTP communication standards and OGC geospatial interoperability standards, so there is no restriction on hardware or software on the client side. The model services can be hosted on servers with High-Performance Computing (HPC) capability to transfer the computing pressure from clients to servers; it also allows model users to run computationally intense models from low performance clients (Lee and Percivall, 2008).

2) Geospatial data services

It is usually time-consuming to identify, request, and receive data to meet the need for running a geospatial model. In addition, running geospatial models may require hosting large amounts of data; this might limit the use of particular models because the data requirement may easily exceed clients’ local storage capacities (Lin, 2008). Meanwhile, the number of large geospatial data centers is increasing on the Internet (e.g., GLCF (http://www.landcover.org), DayMet (http://www.daymet.org), and Geodata.cn (http://www.geodata.cn)), which provides intelligent data querying and accessing services. Using the data services can reduce the need to transfer or store data on clients’ computers (Foster et al., 2008), and makes it possible for clients with modest storage capability to run data intensive simulations.

3) Users and clients

Desktop, single-tier, and multiple-tier systems usually provide command-line or graphic user interface to facilitate human to machine interactions. While the systems highlight human use, they can hardly be coupled with other machine systems to support sophisticated applications that require functionalities beyond any individual system. In contrast, the SOA model and data services provide Remote Procedure Call (RPC) functionalities for collaboration with machine systems. The clients for the model and data services can be autonomous machine systems that remotely collaborate with the services following the interoperability standards. By integrating the services, the client systems can provide different interactive interfaces (e.g., console utility commands or Web browser-based DSS) to human users. The services also can be coupled with other services to construct service chaining (Alam, 2003). The wetland simulation system (see next entry for details) demonstrates a customized client system to the model and data services.

4) Wetland simulation system

The system implements complex procedures to forecast changes in ecosystem services. Disciplinary models from distributed model services are identified and integrated into the system to perform desired model simulations. Instead of building its own data warehouse, the system accesses some model input data dynamically through distributed data services. A map-based user-friendly interface is provided by the system to facilitate usage across the Internet.

2.2. Model sharing concepts

The geospatial model modules (see Fig. 2) are executable programs that implement the model algorithms; geospatial model processes (see Fig. 2) are program instances created from the model modules to perform the model calculation on inputs from model clients. Since model clients may call the model with their own inputs, the model processes may perform different calculations according to the inputs; therefore, each model process executes separately from other processes. Compared to simply sharing model module copies, model services share the calculating performances of model processes to model clients through the Internet. To realize such a design, it is critical to propose a standardized interface for the model services to satisfy geospatial model sharing and accessing.

2.3. Geospatial model service interface

WPS, whose design is based on the Extensible Markup Language (XML), defines a generic interface and a means to provide geospatial-processing capability through Web services (Open Geospatial Consortium, 2007). The generality of the interface is necessary to ensure that standards can be followed by variant practice needs. Meanwhile, the WPS also allows development of application profiles for specific applications by adding additional rules (e.g., requirements on data types and formats). Despite the added rules, the application profile would still comply with the standard, and should be acceptable by other WPS compliant applications.

Ecosystem disciplinary models usually have explicit requirements on model inputs and outputs; therefore, it is important to clearly define data structure, format, and semantic requirements to avoid confusion (Lee and Percivall, 2008; Reitsma et al., 2009). Based on the OGC WPS standard, we put forward Geospatial Model Service Interface (GeoMSI), which is a WPS profile that is customized in terms of sharing ecosystem disciplinary models. Additional rules (e.g., parameter qualifiers) have been added to the profile to reduce the ambiguity of using the ecosystem disciplinary model from a model service. We present examples on accessing shared models using GeoMSI in Section 3.2.
2.3.1. Model service metadata

The model service provides metadata for model clients to understand and access model services in a standardized way (Grimshaw et al., 2009). We developed the metadata schema for model service and process based upon the WPS metadata schema (Open Geospatial Consortium, 2007):

1) Model service metadata provides a general description of the model service (e.g., service identifiers, contactor information, and allowed operations) and a list of the model processes available through the service.
2) Model process metadata provides detailed information on specified processes (i.e., process identifier, title, description, inputs and outputs, and optimized options).

2.3.2. Service operations

The model client interacts with the model service through three predefined WPS operations (i.e., GetCapabilities, DescribeProcess, and Execute) (Open Geospatial Consortium, 2007). By invoking the Execute operation, an interactive conversation is established between the service and the client. Asynchronous calls are supported by the model service due to the fact that the ecosystem disciplinary models calculation could take several minutes or even hours to finish.

2.3.3. Parameter qualifier

We used the Geography Markup Language (GML) 3 data format for exchanging geospatial data in our application. GML 3, the latest geospatial data exchanging standard from the OGC, is platform-independent and supports many geospatial data types including vector and raster. Supporting GML also allows the model services to accept data from distributed geospatial data services, such as the WPS and the WCS services (Open Geospatial Consortium, 2005). We added several qualifiers to avoid or reduce semantic misunderstanding of model inputs/outputs, including:

1) We adopted an open geodetic parameter set compiled and disseminated by the European Petroleum Survey Group (EPSG) to coordinate parameters for geospatial data. EPSG has compiled the most commonly used projections to build their database (Open Geospatial Consortium, 2008). The EPSG identifier is succinct and explicit, and likely promotes geodetic understanding between model services and model clients. The EPSG code can be added to the GML data tag attribute, rather than embedding entire parameters of the coordinate reference system (CRS) description (e.g., Well-Known Text CRS).
2) Metadata references (usually a Uniform Resource Locator (URL) reference to access the full metadata) are embedded in the GML datasets. The URLs are simplified and uniform compared to the full metadata content, which usually have complex hierarchical structure; therefore, using URLs can avoid embedding the metadata content into datasets. Moreover, the metadata URL still provides clues for users to retrieve the full metadata and performs further data mining analysis (Egenhofer, 2009).
3) Data can be categorized as Nominal, Ordinal, Interval, and Ratio (Stevens, 1946). The Interval and Ratio data types should have explicit measurement units to avoid misunderstanding. A Unit of Measure (UOM) value should be added to the Interval and Ratio model inputs and outputs. However, some measurement units are inherently problematic and converting them from one measurement unit to another is inefficient. To address this problem, the model service encourages that all parameters use International Units.

2.3.4. Service request protocols

Although model service interfaces are not constrained to a specific Internet communication technique, we implemented both the SOAP and Representational State Transfer (REST) protocols for the model services. The SOAP and REST are open and widely supported by current systems and tools (Guan et al., 2006). Additionally, they are complimentary. The SOAP is best suited for heavy-weight applications, such as desktop- or server-based model applications, while the REST is more suitable for light-weight applications such as browser-based applications. Supporting both protocols will facilitate various user cases.

2.4. Model sharing platform

2.4.1. Geospatial model sharing platform

Based on our observation, current developed geospatial models barely support the WPS and GML. In order to implement the GeoMSL, current model modules have to be modified to support the WPS and GML, which would involve software engineering, network communication, security, and other tasks beyond the core model...
development (Feng et al., 2009). The Geospatial Model Sharing Platform (GeoMSP) is developed to help modelers reduce redundancy in model implementation, make model development more efficient, and alleviate modelers from the necessity of understanding the details of networking technology. The platform we designed uses a Model-View-Controller (MVC) pattern (see Fig. 3), and includes three modules:

1) Model Integration Module. This module collects metadata from model modules and manages the models to perform calculations. All model modules are plugged into the platform through an application programming interface, called the Geospatial Model Programming Interface (GeoMPI), rather than implementing the GeoMSI.

2) Model Service Module. This module accepts operations defined by the GeoMSI from model clients using the supported service request protocols, such as SOAP and REST.

3) Controller Module. This module dynamically monitors and manages all model processes to ensure efficiently using server resources (e.g., computing, memory, storage resources). HPC techniques, such as grid computing (Bird et al., 2009), can be introduced into the model process through this module. This module also provides utility functions, such as translating a dataset to exchanging data format (e.g., GML).

We developed a geospatial model-sharing platform using the Java 2 Platform Enterprise Edition (J2EE). As one of the Object Oriented Programming (OOP) languages, Java has enhanced network capabilities and numerous libraries that are available to support the platform we developed. Additionally, open-source libraries (e.g., GeoTools (http://www.geotools.org), GeoServer (http://geoserver.org), and OpenLayers (http://openlayers.org)) have been used to empower the geospatial related features (e.g., geospatial data reading and writing, spatial data processing) of the platform. The platform is operating system (OS) independent and can be deployed on any OS that supports the Java Virtual Machine (e.g., Windows desktops, Linux servers).

2.4.2. Geospatial model programming interface

The GeoMPI defines programming rules (i.e., a Java interface, metadata labeling annotations) to regulate behavior of each model module. Each individual model module should follow the rules to share their model calculations on the GeoMSP.

Many OOP languages, including Java, allow programmers to define an interface as a contract between its implemented classes and the outside world. Implementing an interface allows a class to become more formal about the behavior it promises to provide (Thirunarayan, 1999). Java interface Calculate has been defined in the GeoMPI to regulate the behavior of each disciplinary model modules to the GeoMSP. To share the models on the GeoMSP, each model module will implement the functions defined in Calculate. The functions include:

1) calculate: starts the model calculation.
2) getPercent: returns a percent float indicator (from 0 to 100.0) indicating the proportionate progress of current model calculations.
3) getStatus: returns the status of the current model calculation (running, suspended, and stopped).
4) cancel: stops the current model calculation process.

Note that the functions design enables the GeoMSP to provide asynchronous model accessing for model clients, as mentioned in the GeoMSI section. Once a model request is received, the GeoMSP creates a model process from the requested model module and runs the calculate function of the model process in a separate thread. Later on, the GeoMSP checks the status of the model process by calling its getStatus and getPercent functions repetitively, and retrieves the model results when the model process finishes calculation. While the model process is running, the GeoMSP can call the cancel function to abandon the calculation.

Exchanging geospatial data is a routine activity for running geospatial models. Ecosystem disciplinary models may require different data types (e.g., vector feature, raster coverage, and non-geospatial values). In order to address data interoperability across models, the GeoMPI supports the following four data type categories:

1) Primitive data types, such as String, Integer, Float, and Double.
2) Geospatial geometry types defined in OGC abstract specification (i.e., Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and Raster) (Open Geospatial Consortium, 2008).
3) Feature and FeatureCollection defined in OGC abstract specification (Open Geospatial Consortium, 2008).
4) Complex data types are also supported to meet the need for non-geospatial complex structure data exchange, such as Array and Map types (Stephanie and Huang, 2005).

The GeoMPI defined a way for models to provide their metadata. Usually, metadata are saved in text files, which are separated from
the executable model module and provide legible descriptions for model users (Matott et al., 2009). This separation of program codes with model metadata makes it error prone because painstaking effort is needed from modelers to ensure that the metadata files are always synchronized with model modifications. Binding the metadata directly to model classes and parameter functions can eliminate human errors. It can also facilitate metadata retrieval from the models and parameters to reduce the need from human inputs, making it more efficient for modelers to maintain their models. Java annotations provide a way to label metadata information onto either classes or functions and to link metadata with model modules. The GeoMPI uses three annotations to support labeling model classes and parameters with metadata:

1) GeoProcess labels metadata on model class.
2) GeoInput labels metadata on input parameters.
3) GeoOutput labels metadata on output parameters.

2.4.3. Model programming interface implementations

The platform requires each individual model to implement the CalculP Java interface and label model parameters using the metadata annotations. Considering the different programming languages and their execution environments (e.g., Java, C/C++), metadata annotations may be adopted for developing model modules, modelers could implement the GeoMPI in one of the following ways:

1) Direct Implementation. Using this approach, the model module implements the GeoMPI directly and runs in the same execution environment as the platform does. This approach is expected to provide better performance than the adapter implementation approach (see next entry for details). However, models usually have to be modified and the modification may be difficult without access to the source codes of the models.

2) Adapter Implementation. Using this approach, an adapter module is developed to implement the GeoMPI for the model module and connect the model module to the GeoMSP. The adapter module translates model calls and inputs/outputs from the GeoMSP form to the form supported by the model, and it avoids adapter module translation usually requires transforming input and output data back and forth between the model process and the process where its adapter module is running, which slows down the performance.

3. Sharing geospatial models

3.1. Model sharing implementation

Based on the platform introduced above, various ecosystem disciplinary models can be shared on model services. To demonstrate this approach, we implemented five models of diverse ecosystem processes on the platform to illustrate the distributed model calculation capabilities.

3.1.1. Evapotranspiration model

Evapotranspiration (ET) is one of the key processes affecting the hydrology of wetlands. The model was derived from the Penman–Monteith equation (Allen et al., 1998), and estimates daily average ET at a given geographic location and date. The inputs are geographic location and daily meteorological observations (i.e., daily vapor pressure deficit, daily temperature, daily wind speed, daily solar radiation, and albedo). The output is the estimated ET for the specified location and date.

3.1.2. Land terrain model

Elevation is needed to calculate wetland hydrological variables such as water table depth and water surface area. The model estimates the elevation at a given location or region. The Digital Elevation Model (DEM) can be derived from various sources including the Shuttle Radar Topography Mission (SRTM) or Light Detection And Ranging (LiDAR) data. The output can either be a value representing the elevation of a point or a raster dataset representing the DEM distribution in the given region.

3.1.3. Wetland water table model

In order to dynamically simulate the dynamics of water storage, water surface area, and the impacts on other ecosystem services in wetlands, it is critical to constantly estimate the water table depth at the deepest point in a wetland. Once the deepest depth has been estimated, the water table depth at any location in the wetland and water surface area can be estimated based upon the DEM. The water table model estimates the deepest water level on the given date for a given wetland using the following input data: daily precipitation, ET (calculated from the ET model), wetland location, DEM (calculated from the terrain model), and water depth on previous day. Groundwater flow was minimal in the region (Winter and Rosenberry, 1998), and therefore, ignored in this study. For simplicity, we also assumed that runoff occurred after soil saturation.

3.1.4. Water surface extent model

Water surface extent in wetlands is a key controlling factor for many ecological processes and properties. For example, some models (e.g., Cowardin et al., 1995) used it to estimate the number of breeding birds. The model estimates water surface area for a given wetland using the following inputs: wetland catchment boundary, DEM (calculated from the terrain model), and water depth (calculated from the water table model).

3.1.5. Waterfowl estimation model

The PPR supports more than 50% of North American migratory waterfowl (Gleason et al., 2008). To illustrate the utility of above-mentioned models or services in predicting waterfowl dynamics, we implemented the rule-based model developed by Cowardin et al. (1995) to estimate the number of breeding pairs for 12 waterfowl species according to the wetland water surface area (calculated from the water surface extent model) in May of each year.

The five models use both geospatial and non-geospatial data as input or output parameters, and are suitable for demonstrating and evaluating our concept of model sharing via SOA services. Different GeoMPI implementation approaches (See 2.4.3) were chosen for each model based on the editability of the model. The ET model was originally written in C language and compiled into a binary executable. In order to reuse the existing ET model, the adapter approach was used to implement the GeoMPI, and an adapter module was developed to connect the GeoMSP and the ET model. The ET model runs as a console command, and the adapter module prepares inputs as required by the ET model and calls the model command. When the command finishes a calculation, the adapter module wraps the outputs and sends them to the GeoMSP.

The other four models were written in Java. We were able to implement the GeoMPI for them using the direct approach. The codes below demonstrate how we implemented the GeoMSI on the water surface extent model. The model class (WaterRegionModel) implements its own algorithms in the functions defined in the Calculate Java interface. The GeoProcess annotation was labeled on the class to provide metadata for the model.
@GeoProcess(
    title = "Water Surface Extent",
    keywords = {"water surface", "geospatial extent"},
    description = "The model estimates water surface area from wetland bathymetry, catchment boundary, and current wetland water level within the catchment...")

public class WaterRegionModel implements Calculate {
    ...
}

Model input parameters are labeled using the GeoInput, while output parameters are labeled using the GeoOutput. The code listed below shows two input parameters (Catchment and WaterLevel) and one output parameter (WaterRegion). Parameter metadata (e.g., parameter title, supported units, and projection) were also labeled on each parameter.

@GeoInput (title = "Catchment", srid = {"EPSG:4326"})
public void setCatchment(MultiPolygon catchment) {
    ...
}

@GeoInput (title = "Water Level", units = {"meters"})
public void setWaterLevel(double waterLevel) {
    ...
}

@GeoOutput (title = "Water Region", srid = {"EPSG:4326"})
public FeatureCollection getWaterRegion() {
    ...
}

After implementing the GeoMPI, the WaterRegionCal model code was compiled into a standard Java JAR package and deployed to the GeoMSP by copying the package into a folder that has been specified by the GeoMSP as the model deployment folder. The GeoMSP loads the model metadata from each model package through the GeoMPI and shares their calculations processes as model services.

3.2. Accessing shared models

After deploying the models on the GeoMSP, model clients can then call these models following the WPS operations across the Internet. Before calling the models, model users should identify and understand the available models on the model services; parsing the metadata return from the GetCapabilities operation could do this. Fig. 4 shows an example of the model services. The metadata presents the metadata elements as a hierarchical structure using an XML format, including model service information (the Service-Identification element), service contact information (the Service-Provider element), available operations (the OperationsMetadata element), and available model processes (the ProcessOfferings element) (Open Geospatial Consortium, 2007). Although the metadata returned from the GetCapabilities operation can help model clients understand the model, the information is insufficient for accessing the models practically. Detailed model process metadata can be obtained from the DescribeProcess operation. Fig. 5 illustrates the model process metadata for the water surface extent model.

Model services allow model clients to run a given model on the model service by calling the Execute operation. Because model services are shared using open standards (e.g., OGC WPS, GML), any tool or system that complies with those standards can be used to access the models. To demonstrate the utility of integrated distributed model services, we developed an application program using the GeoTools to call the water surface extent model to simulate water surface extent changes when the water table is 0.1 m, 0.5 m, and 1.0 m (see Fig. 6).

![Model service metadata returned from calling the GetCapabilities operation using a Web browser.](image-url)
4. Integrated wetland ecosystem services simulation

The PPR is an area where mid-continental climate variations interact with glacial geology to produce one of the most productive ecosystems in North America, both of agricultural crops and of wildlife. The PPR stretches from Alberta, Saskatchewan, and Manitoba in Canada to Montana, North Dakota, South Dakota, Nebraska, Minnesota, and Iowa in the United States. The PPR is approximately 900,000 km² (Mann, 1986; Pospahala et al., 1974) and may have contained over 20 million ha of wetlands prior to European settlement (Millar, 1973; Tiner, 1984). Soils in the PPR are fertile and the area has been extensively developed for agriculture. Consequently, over 50% of the wetland area in the PPR of the United States (Tiner, 1984) and 71% in Canada (Lands Directorate, 1986) has been drained for agricultural development. Prairie wetlands also are of considerable ecological value and support more than 50% of North American migratory waterfowl and they provide numerous other ecosystem services (Gleason et al., 2008) such as climate change mitigation and water storage. Because competing land use has highly modified this landscape, we choose the area to demonstrate an application of shared open geospatial models to simulate hydrological and ecological change. Moreover, the rise and fall of water due to natural interannual climate cycle influences the provisioning of essentially all ecosystem services in the wetlands and uplands of the PPR.

We developed a dynamic wetland ecosystem simulation system for the PPR. For simplicity of presentation, we concentrated on only a few ecosystem services (e.g., water storage and waterfowl). We are in the process of adding more components to the system to simulate more ecosystem processes and services. Nevertheless, the described approaches in this paper covered all the techniques.

Fig. 5. Input parameters section of the water surface extent model metadata returned from calling the DescribeProcess operation.

Fig. 6. The customized application illustrates dynamic water surface area change at three deepest water table depths, which were calculated by calling the shared water surface extent model.

Fig. 7. Time-driving dynamic wetland ecosystem services simulation model chain.
required for the expansion. In this study, models required to simulate the ecosystem processes of PPR wetlands were integrated into a time-driven model chain for the model services. The model chain simulates major ecosystem processes in individual wetland catchments (see Fig. 7). Models that use different temporal inputs can be included in the model chain. On a daily scale, daily meteorological observations, geographic data, and model outputs from previous day were used as inputs for the model chain calculation. The model chain executes the models from model services and retains model outputs at current time step. Once the model chain finishes daily simulations for one year, it calculates yearly statistics from daily simulation results (e.g., largest water surface area of the year). Finally, the waterfowl estimation model is invoked to estimate the waterfowl breeding pairs of the year based on the water surface area in May.

The model chain retrieves meteorological data directly from DayMet service, provided by U.S. National Center for Atmospheric Research (http://www.daymet.org), and loads geographic data dynamically from a WFS service we developed for this application. We also developed a user-friendly Web site based upon WebGIS technologies to help users access the simulation system. Several libraries from open-source communities have been used in the system, including OpenLayers, Dojo (http://www.dojotoolkit.org/), ExtJS (http://extjs.com/), and Google Chart (http://code.google.com), to facilitate visualization and user interactions. Users can simulate water table depth changes, surface water extent changes, and potential waterfowl breeding pairs for wetlands of interest during certain time period (e.g., day, month, and year) and then display the results interactively (see Fig. 8). Users also have the option of downloading simulation results for further analysis.

5. Discussion and conclusion

The architecture we proposed in this paper presents a distributed framework to develop ecosystem disciplinary models shared and accessed through the Internet. Compared to traditional sharing of model copies, sharing models as model services makes the models accessible to users anywhere in the world, and dramatically expands their applications. By complying with open standards (e.g., W3C, OGC, and ISO standards), the model and data services can be used regardless of the hardware, software, and implementation differences behind the services. The open standards help reduce the interoperability problems that might be encountered when using closed standards, such as commercial private standards. Most open standards (e.g., GML, WPS) adopted in the architecture are based on XML. Although the XML-based standards can be easier in
parsing and understanding compared to binary-based standards, they tend to increase data transmission through the Internet, which may slow down the performance especially when heavy data transmission is involved. Further research is needed to investigate the performance of standards and propose possible optimization procedures. Since the service-oriented concept is also fundamental for other distributed computing architectures (e.g., GRID computing, cloud computing) (Grismshaw et al., 2009), the data and model services paradigm proposed in this study should have great implications to develop high-performance computing capabilities.

Based on the OGC WPS standard, we developed the GeoMSI to provide an explicit service interactive interface for sharing accessing geospatial ecosystem models through model services. Besides the operational and metadata defined by the WPS, we added rules to specific input/output model parameters (e.g., measurement unit, coordinate reference system) for model applications. One of the main challenges for modelers to make traditional geospatial models sharable through the GeoMSI compliant services is that they understand the technology and develop appropriate services for the models. The GeoMSP developed in this study can handle issues related to establishing the geospatial model service, allowing modelers to implement the GeoMPI programming interface without directly developing model services and, therefore, focus on model algorithm implementation. Other modeling frameworks (e.g., the CDSM, the ESMF, and the OpenMPI) have proposed programming rules to dig inside the model structure and use the functionality to extract the most successful pieces from models (Voinov et al., 2008). The approach proposed in this paper has highlighted the interoperability issues for model sharing in a distributed environment; further research is needed to investigate how to use this approach in other modeling frameworks to address the interoperability difficulties in these frameworks, especially in an Internet-based computing environment.

The integrated wetland ecosystem modeling system introduced in this paper implemented a time-driven wetland ecosystem process simulation based on integrating shared models and data from remote services. The system demonstrates the benefits of openly sharing disciplinary models for integrated interdisciplinary ecosystem simulation. Wetlands are one example of an ecosystem where interdisciplinary problems are the norm where the need to promote synergy among disciplinary models to address complex ecological phenomenon remains. Such applications can include complex workflows for data analysis and simulation tasks for any equally diverse set of users. Sharing and integrating scientific models as network services is an important approach to meet these needs. In this study, we have identified methods for sharing wetlands geospatial computational models as model services. These methods have clear advantages over standalone or closed network modeling systems on model accessibility and interoperability, and can be easily applied to other model development activities. Model sharing not only helps modelers share their models, but also makes it possible to capitalize on abundant scientific resources available worldwide to solve more sophisticated ecosystem simulation problems.

Acknowledgements

This study was funded by U.S. Department of Agriculture (USDA) National Resources Conservation Service’s (NRCS) Conservation Effects Assessment Program (CEAP) – Wetlands, USDA Farm Service Agency (FSA), USGS, State 863 Program of China (2006AA01A120), and Natural Science Foundation of China (40801180). Zhengpeng Li provided the ET model module. The authors also give special thanks to Craig Walters for improving the English.

References

Leavesley, G.H., Markstrom, S.L., Brewer, M.S., Viger, R.J., 1996. The modular modeling system (MMS)—the physical process modeling component of a database-centered decision support system for water and power management. Water, Air, and Soil Pollution 90 (1), 293–317.