Model Sharing based on Distributed GIS

A Case Study of Tapes-G Model

Min FENG
Department of Biological
Graduate University of Chinese Academy of Sciences
Beijing, China
fengm@lreis.ac.cn

Yunqiang ZHU
Institute of Geographical Sciences and Natural Resources Research
Chinese Academy of Science
Beijing, China
zhuyq@lreis.ac.cn

Abstract—**Distributed Model Sharing Service provides a platform for GIS models to share their functions. The service is developed under the .Net environment and runs as Web Services. The interface of the service is designed, which conforms to the specifications of OGC, such as OWS and WPS. Therefore, service requestors will not be limited to any operation systems and software. As an example, the work of sharing Tapes-G, a topical hydrological model, is introduced.**

Keywords- Distributed GIS; Computer Model; Tapes-G; Models Sharing

I. INTRODUCTION

Distributed Internet technique is very important for Geographic Information Systems (GIS). There are lots of scientific compute models which are the fruit of long-term research and practice, but most of them can only run at isolated system or special platform [1]. Thus it is very urgent and interesting for transforming those models to share their functions on Internet.

There are several different distributed compute platforms put forward since 90s last century, such as DCOM, CORBA, J2EE [2]. Some researchers have built distributed GIS based on those platforms [3]. However, those systems have no interoperability on different distributed compute platforms because each platform they used is not opened to others. For example, DCOM was developed by Microsoft and CORBA was brought forward by OMG, and they can neither exchange information nor invoke the services provided by the other directly [2]. Therefore, this paper adopts Web Services as the distributed compute platform of the Model Sharing Service. Web Services is a specification recommended by W3C in 2002, and it is independent with both the operation system and the development environment.

The interactive interface of service is designed according to WPS (Web Processing Service Specification) and OWS (OGC Web Services Common Specification). WPS and OWS are parts of the specifications which are recommended by OGC. Implementing the interactive interface conforms to WPS can help to reduce the obstacles of interoperability between service and client systems, and the functions of the model can be accessed by any client systems that supports WPS [4][5].

In the Model Sharing Service, GML is adopted as the data format for transporting GIS data between distributed servers and clients. There are many GIS data formats, such as ESRI Shapefile, ESRI Coverage and MapInfo TAB; however, those formats are not suit to be used on Internet, because they are composed of several binary files, and are supported by special system. GML is brought forward by OGC in 2002, and the last version is 3.1. Similarly as Web Services, GML is also designed based on XML; therefore, any GML document can be exchanged as a string block, and not restrict by special operation system [6].

Tapes-G (a Grid-based Terrain Analysis Program for the Environmental Sciences) is a typical hydrological compute model which was developed by the Australian National University. The model provides functions of computing terrene indexes and radiance indexes [7]. In this paper, we publish Tapes-G on the distributed model sharing system to demonstrate the detailed process of sharing model on Internet.

II. ARCHITECTURE

A. Architecture

The architecture of distributed model sharing is based on the publish/find/bind pattern shown in Figure 1, and it supports the dynamic binding between service providers and requestors. There are five essential roles in the architecture [8]:

- *Model Sharing Service* provides a process sharing platform for models. It can be published to the Service Register.

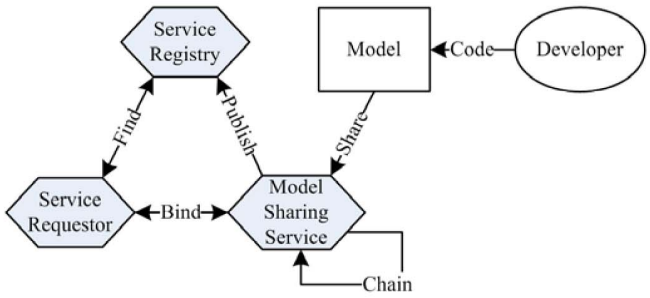

Figure 1. Architecture of Distributed Model Sharing

The research is funded by Earth System Scientific Data Sharing Program (2004DKA20180)

- *Service Registry* acts as a registry or clearinghouse of services that helps service providers and service requestors to find each other.
- *Service Requestor* is user of model services. It finds the model services it desired on the Service Registry and then accesses the model services from their providers.
- *Model* is a concrete program or arithmetic that implements special functions. A model can share its functions on Internet by plugging into the Model Sharing Service.
- *Developer* is owner of models. Developers develop and establish concrete models to satisfy the requirement of service requestors.

Web Services are not only a protocol for exchanging information on Internet, but also are footstone of the architecture of distributed compute platform [2]. Thus, the Model Sharing Service takes advantage of the Web Services to form its own distributed compute architecture.

B. *Model Sharing Service*

Model Sharing Service shares the functions of models to the requestors based on Web Services protocols, so the requestor can invoke the operations of the interface by SOAP (Simple Object Access Protocol). The interface of the service is designed according to WPS, and provides three operations in the interface, i.e. GetCapabilities, DescribeProcess and Execute as shown in Fig.1. Each operation represents different level of information about sharing and accessing functions of models.

WPS has two predefined encoding format for the input and output parameters of the operation, i.e. XML (Extensible Markup Language) and KVP (Key Value Parameter) [4]. KVP is mostly used with HTTP GET, and is not suit to be transported using SOAP. Thus in WPS, these parameters are all encoded with XML and their structures are defined using XML Schema. In order to simplify the schema structure, these parameters of operations are defined as "string" instead of predefined structures or classes. The above expression method need the server and clients to parse and validate parameters' XML strings, which is more bother than using predefined structures directly but can greatly reduce the complexity of using WSDL (Web Service Definition Language) to describe the model service.

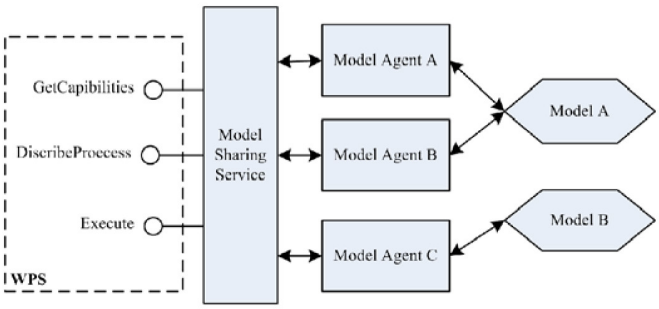

Figure 2. Architecture of Model Sharing Service

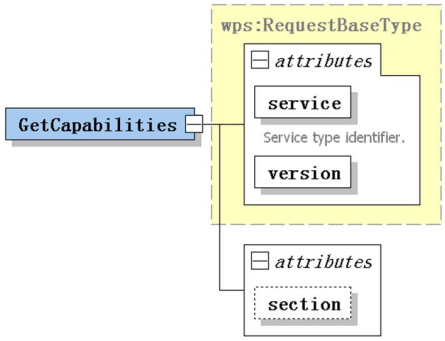

Figure 3. Structure of "GetCapabilities" element

1) *GetCapabilities*

This operation allows a client to request and receive service metadata documents that describe the abilities of the specific model service. Since service metadata is the basic information for any service, the GetCapabilities operation is required for any service [5]. The structure of the GetCapabilities operation is shown as Fig.3.

WPS does not provide the XML input encoding structure, thus a structure is defined by us. The "GetCapabilities" element extends from "wps:RequestBaseType", and "section" attribute is added to the element which is used to indicate the sections the client want to get. There are four sections provided by "GetCapabilities" operation:

a) ServiceIdentification: Return general metadata from specific server.

b) ServiceProvider: Return metadata about the organization that provides this specific service instance or server.

c) OperationsMetadata: Return metadata about the operations and related abilities specified by this service and implemented by this server, including the URLs for operation requests.

d) ProcessOfferings: Return the brief description of the processes offered by this server.

2) *DescribeProcess*

This operation allows a client to request and receive detailed information about one or more processes including the input and output parameters and their formats. This description can be used to automatically build a user interface capturing the parameter values to execute a process instance.

Each parameter is described by a data structure that specifies the allowable formats, encodings, and units of measure (when applicable). There are three types of parameter data:

a) ComplexData: this type can be any form of information, such as GML, XML, Base64 encoded data, etc. Each ComlexData element has three attributes to describe the data, i.e. format, encoding, and schema. "format" identifies the input or output format supported using MIME (Multipurpose Internet Mail Extensions), such as text/xml. "encoding" identifies the encoding of the data, such as UTF-8,

UNICODE, etc. "schema" identifies the schema of the data, such as the url of a XML Schema document.

b) LiteralData: this type indicates that the input parameter shall be a simple literal value (such as an integer) embedded in the execute request, and it also describes the possible values.

c) Bounding Box information: this type indicates that the input parameter shall be a BoundingBox data structure. In addition, it provides a list of the CRSs supported in these Bounding Boxes.

3) Execute

This operation allows a client to run a specified process implemented by the WPS. Noticeably, only after the client provides correct input parameter values, the Execute operation can run correctly and return desired results; otherwise, the exception will be returned from the service.

4) Exception

When the system encounters an error in the process of performing an operation, it shall return an exception report message as specified in OWS. The type of the exception is represented by the "exceptionCode" attribute. The attribute value can be a predefined string in WPS, such as "MissingParameterValue" that means some required parameters are not provided. Clients can parse the code to understand the reason of the exception. Furthermore, "locator" and "ExceptionText" can pass the client more information about the exception. "locator" shall indicate to the client where an exception is encountered. "ExceptionText" describe the exception message by using readability text.

C. Map Agent and Model

Each model has its own characteristics and is different to other models; therefore, models can not be shared directly on the Model Sharing Service. A middleware layer named Model Agent is finally adopted to solve the integration problems between the service and models. Model Agent is a module that wraps special model, and presents identical interface to the Model Sharing Platform. Each model must be wrapped into a specific Model Agent before sharing it on the Model Sharing System. The relationship among Model Agent, the Model and the Model Sharing Service is shown in Fig.4.

The main functions of a Model Agent include:

- *To describe the model it wrapped.* Model Agent provides the Model Sharing System with necessary information about the model and the processes.
- *To interact with the model.* Model Agent has to load and invoke the model in right way. It passes the input parameters to the model, and returns the results to the Model Sharing Service. Each model may be different, such as the type and encoding of the input and output parameters and etc., so Model Agent must be able to erase those different for Model Sharing Service.

Each Model Agent should be compiled as a .Net Assembly, and implement the "IModel" interface. Model Sharing Service loads the Model Agents from Model Repository and communicates with the Model Agent via "IModel" interface.

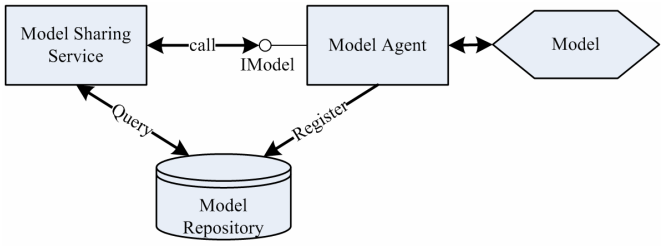

Figure 4. Relationship of the shared model

"IModel" interface has only two operations as follows:

```
public interface IModel
{
    string GetDescription();
    string Execute(string p);
}
```

- *"GetDescription":* return the description of the processes of the model. A model may offer more than one process; therefore, this operation returns the description of all processes the model offered. The output string is formatted by XML and accords to "wps: ProcessDescriptions".
- *"Execute":* is used to invoke specific process of the model, and the input and output parameters is formatted by XML and accord to "wps:Execute" and "wps:ExecuteResponse".

D. Service Registry

Model Sharing Service is carried out as a Web Service; therefore, it can takes advantage of the register technique of Web Services. There are many techniques that can be chosen as the service registry, such as UDDI, WS-Inspection, DISCO, etc [2]. UDDI is a technical specification for describing, discovering, and integrating web services. UDDI registry is a "meta service" for locating web services by enabling robust queries against rich metadata. WS-Inspection and DISCO are much similar, for they are both file based service register specification. UDDI is a powerful pure service registry, and the file based register can be used as complement of UDDI. We established a UDDI registry using Windows 2003, and registered the service on it.

E. Service Requestor

Supporting SOAP and XML is the basic request of the client of Model Sharing Service. There are many programming languages and developing platforms to support Web Services, such as .Net, J2EE, python, etc. Thus we can take advantage of them to interact with the distributed Model Sharing Service.

III. PUBLISHING TAPES-G

A. Compiling Tapes-G

Tapes-G compiles different functions into different executable programs. For example, Terrain index functions are in the TAPES program, and radiance functions are in the SRAD program. We try to publish the terrain index functions and the radiance functions of the model on the Model Sharing Service.

The model was originally written for the NUIX or LINUX operation system using FORTRAN. It can not be compiled and run on Windows directly because it uses some X11 libraries which are not available for Windows. Finally, Cygwin is selected to solve the problem. Cygwin is a Linux-like environment for Windows, and it provides the tools and libraries that can compile Tapes-G, such as GCC and X11 library, etc. Any program compiled with Cygwin can execute in the Windows environment with an additional DLL file "cygwin1.dll".

B. *Developing Model Agent*

TAPES and SRAD are console programs, and they interact with users by console window. The Model Agent redirects the input and output streams from console window to itself to control the execution flow of the model at runtime.

The input and output data of TAPES and SRAD are files with specific GIS data formats. Although there are 11 GIS data formats can be read by Tapes-G, GML is not supported. Therefore, the Map Agent transforms the input data from GML into the DEM file (ARC/INFO GRIDFLOAT), and then passes the filename to the model. After the model finishes its process, the Model Agent retrieves result information from the output file. Subsequently, the output file is automatically transformed into GML format, and then passed to the Model Sharing Service. The workflow of Tapes-G on the Model Sharing Service is shown in Fig.5.

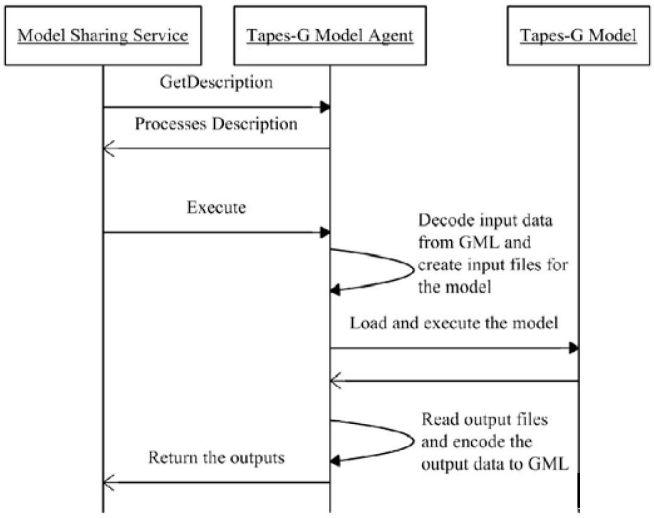

Figure 5. Sequence of invoking Tapes-G processes

C. *Clients*

There are two clients using the functions from the Model Sharing Service. The first client is developed using VBA and execute within ArcMap. This client can transform specific raster data into GML string block, and send it to the distributed Model Sharing Service, and finally show the result in ArcMap. The second client is developed as a standalone python program. This client can process a batch of DEM files, and save the result into specific directory.

SUMMARY

The Model Sharing Service provides a distributed platform for sharing the functions of GIS models. Users can search models they need from the Register Site by using UDDI, WS-Inspection, or DISCO, and invoke the functions following the request of WPS. GML is very suit to be adopted as the spatial data format exchange on Internet. Several models are published on the platform, and we think the platform is a good attempt to share the GIS models on Internet. However, there is still a long way to go. Firstly, more models should be published to test the platform; Secondly, The performance is not good especially for processing big data. Thirdly, to optimize the data stream and control stream using Grid technology.

ACKNOWLEDGMENT

The research is funded by China National Scientific Data Sharing Program of Ministry of Science and Technology of PRC: Earth System Scientific Data Sharing. Especially thanks to the leader of this research: Pro. SUN Jiulin and all of other program members.

REFERENCES

[1] Michael B. Smith, Dong-Jun Seo, Victor I. Koren, Seann M. Reed, Ziya Zhang, Qingyun Duan, etc, "The distributed model intercomparison project (DMIP): motivation and experiment design", Journal of Hydrology 298 (2004), 4-26

[2] Ethan Cerami, Web Services Essentials, O'Reilly, 2002

[3] Fangju Wang, A Distributed Geographic Information System on the Common Object Request Broker Architecture GeoInformatica, 2000, 89–115

[4] Open GIS Consortium, Inc., OpenGIS Web Processing Service, Open GIS Consortium, Inc. 2005

[5] Open GIS Consortium, Inc., OGC Web Services Common Specification, Open GIS Consortium, Inc. 2005

[6] Ron Lake, The application of geographymarkup language (GML) to the geological sciences, Computers & Geosciences 31 (2005) 1081–1094

[7] John Gallant, Terrain Analysis Programs for the Environmental Sciences - Grid, unpublished

[8] Open GIS Consortium, Inc., OpenGIS Reference Model. Open GIS Consortium, Inc. 2003