International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-4/W2, 2013
ISPRS WebMGS 2013 & DMGIS 2013, 11 – 12 November 2013, Xuzhou, Jiangsu, China
Topics: Global Spatial Grid & Cloud-based Services

KEY TECHNOLOGY RESEARCH ON OPEN ARCHITECTURE FOR THE SHARING OF HETEROGENEOUS GEOGRAPHIC ANALYSIS MODELS

S.S. Yue, Y.N. Wen, G.N. Lu, D. Hu

Key Laboratory of Virtual Geographic Environment (Ministry of Education),
Nanjing Normal University, Nanjing, Jiangsu Province, 210046, China -
yss123yss@126.com, wenyn@msn.com, gnlu@njnu.edu.cn, hud316@gmail.com

KEY WORDS: Geographic Analysis Model; Resource Sharing; Distributed Computing

ABSTRACT:

In recent years, the increasing development of cloud computing technologies laid critical foundation for efficiently solving complicated geographic issues. However, it is still difficult to realize the cooperative operation of massive heterogeneous geographical models. Traditional cloud architecture is apt to provide centralized solution to end users, while all the required resources are often offered by large enterprises or special agencies. Thus, it's a closed framework from the perspective of resource utilization. Solving comprehensive geographic issues requires integrating multifarious heterogeneous geographical models and data. In this case, an open computing platform is in need, with which the model owners can package and deploy their models into cloud conveniently, while model users can search, access and utilize those models with cloud facility. Based on this concept, the open cloud service strategies for the sharing of heterogeneous geographic analysis models is studied in this article. The key technology: unified cloud interface strategy, sharing platform based on cloud service, and computing platform based on cloud service are discussed in detail, and related experiments are conducted for further verification.

1. INTRODUCTION

The earth system is inherently a complicated huge system, and involves numerous research fields. According to distinctive characteristics and demands of different fields, researchers have developed abundant geographical analysis models for modelling and simulating various geographic issues. These models are generally heterogeneous in structure, data, and runtime environment. The heterogeneity of geographical analysis models not only leads to difficulties in the distributed sharing of single model, but also brings challenges for the integration and collaborative simulation when solving comprehensive geographic issues.

Different modelling and reuse frameworks have been developed to tackle the environmental problems (Morozov et al. 2006), including the Spatial Modelling Environment (SME) (Maxwell and Costanza 1997a), the Interactive Component Modelling System (ICMS) (Reed et al. 1999), the Modular Modelling System (MMS) (Leavesley et al. 2002), the Earth System Modelling Framework (ESMF) (Hill et al. 2004), the Open Modelling Interface (OpenMI) (Blind and Gregersen 2005), the European Union's Program for Integrated Earth System Modelling (PRISM) (Valcke et al. 2006), and CAPRI (Britz et al. 2010). Although these frameworks are developed with valuable functionality, most of them are highly field-related and platform-related, which limits the breadth of application and increases the entry barriers of these systems. On the other hand, few of these frameworks support model sharing and integration over the Internet (Feng et al. 2011).

Cloud computing is the integration of several advanced IT and ideas, including supercomputing technology, network communication technology, virtualization technology and SOA (Ahrens 2010). The adoption of stratified service mechanisms (e.g. IaaS, PaaS and SaaS) renders cloud computing transparent and open, which will provide the opportunity for model providers and users to focus on their own business at the application level, rather than on how to deploy or execute models. Thus, cloud computing can effectively support the construction of an open environment for the sharing of heterogeneous geographical analysis models.

In this article, the key technology of the proposed open environment for the sharing of geographical analysis models based on the cloud computing architecture is studied. With this environment, model owners can submit models more conveniently and model users can access the submitted models transparently and conduct further execution using computing resources provided by the cloud computing service. The remainder of the article is structured as follows: Section 2 proposes the architecture of the open environment for geographical analysis model sharing and the basic cloud interfaces are studied. The sharing platform is described in Section 3 and computing platform is described in Section 4. Section 5 introduces the realization of the prototype system. Finally, conclusions and discussions are presented in Section 6.

2. UNIFIED CLOUD INTERFACE STRATEGY FOR HETEROGENEOUS MODELS

2.1 Design of the architecture for geographic model sharing

To share heterogeneous geographic analysis models in the distributed web environment, two platforms are necessary no matter what kind of sharing architecture is been employed: (1) sharing platform and (2) computing platform. Figure 1 shows the basic idea of the open architecture this paper designed.

(1) Sharing platform: Model providers would offer or register their model item in the sharing platform, and model user would discover their interested models in the sharing platform. The model sharing platform is somewhat similar with some other resource sharing space (Diaz et al. 2008, Granell et al. 2010, Feng et al. 2011), but the shared model resources are more complicated and flexible. Geographic analysis models contain the author's specific understanding of the geographic problem, and such understanding can be hardly described by metadata which is current widely used in the data sharing field.

(2) Computing platform: After model resources are shared in the open web environment, the next step would be using the

models. But geographic analysis model execution often involves mass geographic data processing and running of complex equations and long-sequence simulation, which requires a large amount of computing resources. Also, these models are developed by different programming languages, depends on specific operate systems, and supported by diverse hardware architectures, resulting the execution heterogeneity of geographic analysis models.

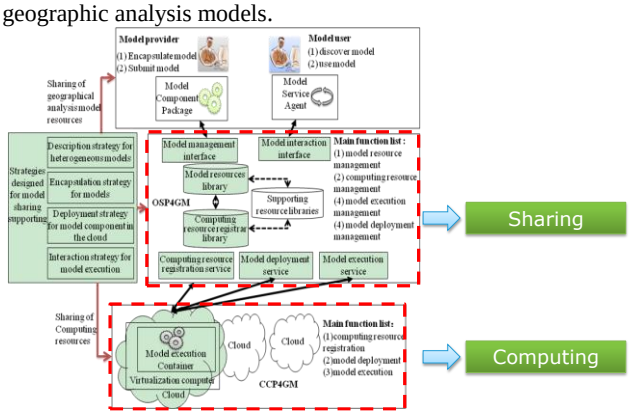


Figure 1 Open architecture for sharing heterogeneous geographic models

To use the model which model provider provided (regardless of the sharing environment), a model user should at least clearly know how to prepare the input data, and what does the output data meaning. Geographical analysis models are the abstractions of geographical information and geographical processes, the data retrieval and processing require knowledge of specific domains. Due to most geographic models are complicated (Crosier et al. 2003, L ü 2011), the model does not only need input data once, every step would ask the model user to provide corresponding data. So the model user should also understand what the middle state meaning in the model running process, before they can correctly using the specific model. Therefore, the input & output data and the model running behaviour are the key technological points of sharing the heterogeneous models in open web environment.

2.2 Strategy of model data description method (UDX)

The input & output data of the model would be the only way model users try to understand the geographic analysis model, when the model is treated as a black box. Due to the complexity of geographical problems and the diversity of modelling methods, different model involves different data. And even to the same model and the same data, as a consequence of the variety of model researchers' view points, the data would be constructed in different format (e.g. plain text format or binary format), in different organization structure (e.g. single file or multi files). In this case, a fixed data model or data format is hard to be created to handle all these model data characters. To solve this problem, this paper designed a flexible description mechanism for the model data.

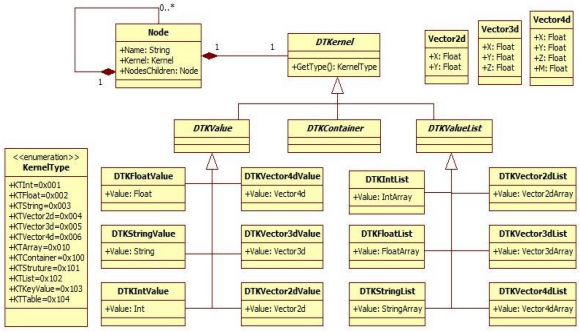
Based on the above analyses, we designed a Universal Data eXchange (UDX) object model. The UDX is not only a simple data model, but should be treated as a sort of data representation model or the meta-model of data model. The base idea of the UDX is to provide basic data construct element, and finally form the data through the combination and iteration of these basic elements. The UDX consist of two parts:

(1) The UDX object model: used to express the data structure;

(2) The semantically enhanced schema: used to attach UDX objects with semantics information.

2.2.1 Universal Data eXchange object model

UDX object model contains *Node* and *Kernel* two basic construction elements. *Node* controls geographic model data's hierarchy structure; *Kernel* controls the specific behaviour of *Node*. Every *Node* has its unique *Name* and *Kernel*, and possesses amount of child *Nodes* (0 to n).



Finger 2: UML diagram of UDX object model

The Unified Modelling language (UML) diagram of the UDX object model is shown in figure 2. Enumeration constant table is defined to give all *Kernels'* type a unique code. And the enumeration can be divided into three groups: *DTKValue*, *DTKValueList*, and *DTKContainer*. The detail kernel type is shown in table 1.

Table 1 *Kernel* type of *Node*

Group	Kernel Type
DTKValue	*KTInt, KTFloat, KTString, KTVector2d, KTVector3d, KTVector4d*
DTKValueList	*KTArray*
DTKContainer	*KTStructure, KTList, KTKeyValue, KTTable*

If a node's kernel type belongs to the *DTKValue*, the kernel should limited to one of the Int type, Float type, String type, and Vector type (Vector2d, Vector3d, and Vector4d). And the node cannot possess any child node, the value that the node contains must be consistent with the kernel type.

DTKValueList kernel indicates the node has a list of value, and the value must be the same type which is one of the kernel type in the *DTKValue* group. Also, the *DTKValueList* cannot possess any child node.

If the node's kernel type belongs to the *DTKContainer*, there four types it can select from: (1) *Structure*; (2) *List*; (3) *Key-Value*; (4) *Table*. The *DTKContainer* node itself does not hold any data value, the specified kernel type indicates what the kernel type its child nodes are, and all the data should be stored in the child nodes.

Thus the Universal Data eXchange object model (UDX) is designed to describe model data for model providers and model users. By using the basic value type and hierarchy (tree) structure, all geographic model data can be easily represented and exchanged.

2.2.2 UDX object and UDX sematic enhanced schema

To geographic model users, the model data need to be expressed more than the data content itself, but also information behind the data content. The designed UDX object model can more completely express the data content and structure, but the semantic and contextual relationships of data in the model execute process are not able to be expressed within it. Although these information can be conveyed between the model provider and the model user by writing an explicit introduction document,

it would make the sharing work more convenient by take advantage of formal methods. And also, the introduction document method is too flexible and uncontrollable to a universal geographic model sharing platform.

Geographic model is tightly related to spatial and temporal scales and semantic information. And along with the development of geographic modelling research, several general data structures have been established and have become a kind of general data model in specific research field. Take Triangulated Irregular Network (TIN) as an example, number of triangle vertex indexes in a TIN must be no greater than the number of points in the point sequences. According to the above analysis, the paper made a summary of these types of information that is vital to the geographic model sharing: (1) Spatial-temporal measurement information: such as the unit and spatial reference system information; (2) Semantic (concept) of data: such as soil porosity and flow velocity; (3) General data representation methods in specific domains: such as DEM; (4) General data constraint information: such as TIN and Grid.

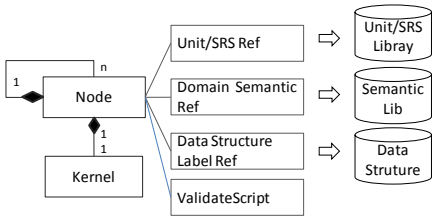


Figure 3 Design of semantic enhanced UDX schema

In this case, based on the design of the UDX object model, this paper designed the UDX self-consistent schema. Figure 3 shows the design of semantic enhanced UDX schema. All these essential information would be attached to the UDX data node using Key-Value mode. Figure 4 shows a typical example semantic enhanced UDX schema, which expressed the Gaussian Diffusion models data content, and the related unit & sematic information.

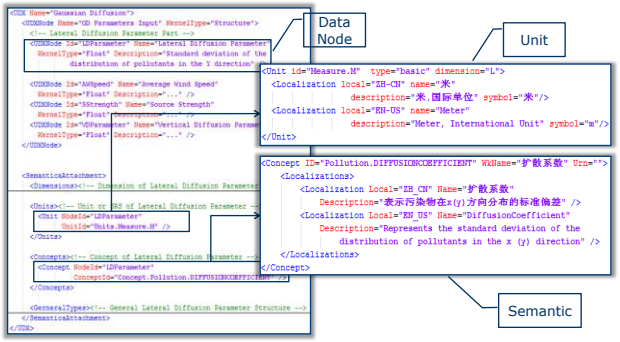


Figure 4 Semantic enhanced UDX schema

2.3 Abstraction of heterogeneous geographic analysis models

The cogitation differences between model providers and model users lead to the difficult of sharing heterogeneous geographic analysis models. As to the knowledge and specialized field are varying in the majority researchers, a model provider can't clearly and detailed explain what the model meaning to a model user. Geographic model execution process can be treated as several geographic states, and the related model data traversed from one state to another state. Model users prepare the input data for the model can be treated as model execution event. Hence, the paper proposed a model behaviour description method: State Machine-Event Response model (SM-ER model). The SM-ER model abstracts geographic model execution process as states, and input-output data as event. The interaction between model execution and model user has been described as *request* and *response*.

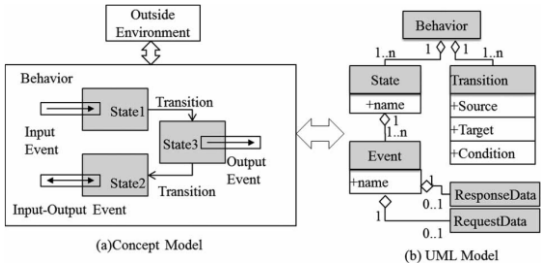


Figure 5 Design of SM-ER model

Figure 5 illustrated the SM-ER model. The geographic models, which can be regarded as progress simulation, consist of a series of states. The *Behavioural* object is the top-level one, which includes *State* sets and *Transition* sets. The *Transition* set describes the source and the target of the state transition and the conditions. The *State* set includes a name and its associated events, which are comprised of three fields: *Name*, *RequestData* and *ResponseData*. *RequestData* is designed to describe a request for data from the outside as an external input; *ResponseData* is used to describe the data submitted to the outside as an output. Therefore, if an Event contains *RequestData*, the model is requesting a user input; if *ResponseData* is contained in the Event, the model is providing data as an output.

Any strict execution order and data requirement can be represented with the SM-ER model, and combined with the UDX model; it can clearly reflect the execution process and benefit collaborative interactions between the models and the external environment.

3. SHARING PLATPORM BASED ON CLOUD SERVICE

According to the analysis in section 2, the geographic model data and behavior interfaces are designed based on cloud service. To shield to understanding differences between model providers and model users, some fundamental resources should be shared communally by these two roles. For the model providers, they should package their geographic models based on the abstraction interface of model behavior that proposed in section 2, thus to support geographic model execute in open cloud environment. And also, the model resource and its depended computing resource should be registered together by model provider in the sharing platform.

3.1 Fundamental resource service

In terms of a specific data content, the *Dimension* is unique no matter what scale *Unit* is been used. Take the length of a segment of road as example. The *Unit* kilometers, feet, or centimeters all can be used, but the *Dimension* can only be Length. Based on the GB-3100-3102/ISO-1000:1992 standard, the *Unit* is divided into *Basic Unit*, *Primitive Unit* and *Compound Unit*. Correspond to the *Dimension* Length, *Unit* meter (symbol m) is the *Basic Unit*, and 1000 meter is *Unit* kilometer (symbol km) that belongs to the *Primitive Unit*. The velocity *Unit* (e.g. km/h) is assembled by two *Units* (kilometer and hour), it belongs to the *Compound Unit*. Figure 6 shows the design of *Dimension* and *Unit* object model. Localization object explains the *Unit* and *Dimension*, and extended localization item can be added to the localization collection according to the description language.

Geographic semantic or geographic concept is been used to describe the meaning of geographic entities and the relationship between these entities. Because of the variety of research field

and the perspective differences among geographic researchers, it is hard to establish an all-purposed geographic semantic (concept) library. The paper do not focus on the semantic library system itself, but try to provide a general semantic construct method, which can load and integrate all existing and already been organized semantic library. Figure 7 shows the semantic object model that the paper designed. The geographic object was abstracted as Concept class, and possessed a unique identification (*id*), a well know name (*wkName*), and a localization collection (*localizations*). The classification system was abstract as Classifier class, and also been provided with a unique *id*, and *localization* collection. Each Concept object belongs to several Classifiers, which can ensure different geographic semantic classification systems be integrated and the information be exchanged lossless.

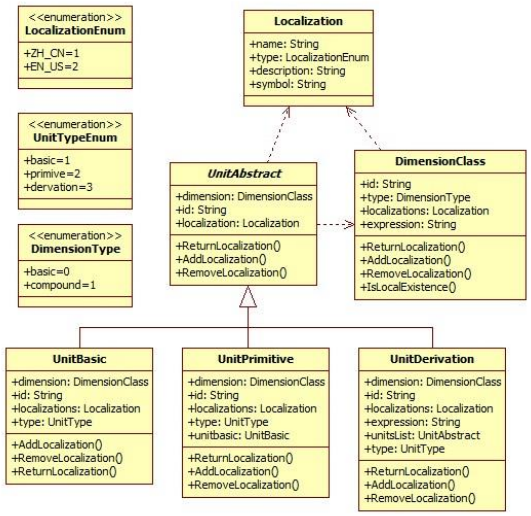


Figure 6 Dimension and Unit object model

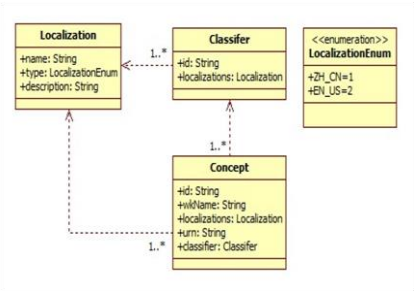


Figure 7 Semantic (concept) object model

3.2 Model and computing resource registration service

Base on the above analysis, the Dimension library, Unit library and Semantic (concept) library were constructed to support all model-related information hided besides the data content be clearly described. Sharing all these fundamental resources is the first step to build an open geographic model sharing environment, and the second step would be sharing the model and computing resources based on the first step. According to analysis in section 2.1, to share all the heterogeneous geographic models, the model data and model behavior information must be transmit from model providers to model users with no misunderstandings. Considering the variety of geographic model execution condition (e.g. system platform, software develop kit environment), the original geographic model should be encapsulated by the model provider (explained in section 4.1), and this paper called the encapsulated model as model component. A model component would be registered into the model library in open sharing platform by the model provider, and the model data and model behavior information should be uploaded with the model at the same time. Besides, the model user would search in the model library with some keywords about their interested models. So the metadata about model component should also be integrated into it. Moreover, the execution purpose for model sharing combined the sharing platform and computing platform as a whole open sharing environment. Model components must be packaged and deployed in the cloud for transparent access, after models are encapsulated into model components. In this case, the Model Description Language (MDL) was proposed in this paper.

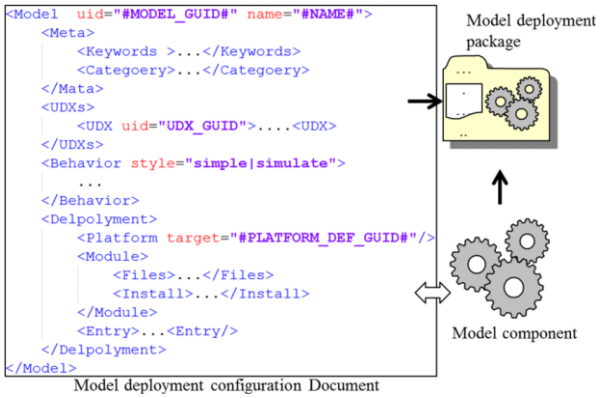


Figure 8 The Model Description Language

Figure 8 shows the structure of the Model Description Language: (1)A *Model* node to contain top-level elements, including a model Identity (ID) name and other information; (2) A *Meta* node to describe metadata information, including the author, classification, keywords and other information, which contributes to the convenient discovery of the models; (3)A *UDXs* node to describe the data required for the model running as expressed in semantic enhance UDX schema; (4)A *Behaviour* node to define the executable behaviour of the model; (5)A *Deployment* node to describe model deployment information. This type of node includes the following child nodes: (a) a *Platform* node to define the executing environment of the model; (b) a *Module* node to describe the module information for the model deployment; and (c) an *Entry* node to describe the entry information to launch the model code.

Thus, all the model-related, data-related and computing-related information could be prepared by model providers, and model users could access these information entirely. Although the MDL document would become somewhat lengthy, both the model provider and model user should not be confused as the encapsulation tool, package and deployment tool were opened to the model providers, and the portal website of the open sharing environment translated the MDL document to a more convenient html webpage for the model users.

4. COMPUTING PLATFOM BASED ON CLOUD SERVICE

4.1 Model encapsulation strategy

Geographic models are programed by model researchers fielded in several different subjects, and the execution calling method and data input/output interaction between model and users both diverse from each other. According to the SM-ER model which is designed to describe model execution behavior (see Section 2.3), this paper developed the universal geographic model encapsulation interfaces. Model providers can encapsulate

model codes into standardized model components based on the definitions of these interfaces.

Figure 9 shows the abstract model of the interface standards, which includes three abstract interfaces: *IModelSessionStub*, the interface of a model session stub for interactions between the model code and the outside environment, which is responsible for releasing events and states during model execution. *IProcedureModelCore* and *ISimulateModelCore* are the last two interfaces, representing the interfaces in simple and complex interaction models, respectively, as described in Section 3.2. Model providers can encapsulate model codes into standardized model components based on the definitions of these interfaces.

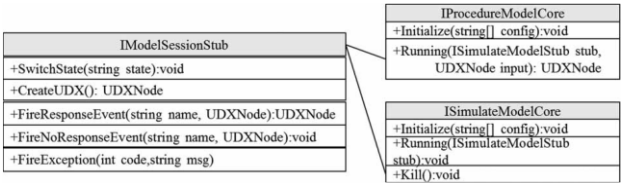


Figure 9 Model component encapsulation interfaces

An object with an *IModelSessionStub* interface communicates with the external environment following the SM-ER model. The definitions of its functions are as follows (with parameters omitted): (1) *SwitchState*. This function is used to notify the external that the model has changed its state. (2) *CreateUDX*. This function is used to create an empty data object expressed by the UDX model, which can be used to transfer data of the model. (3) *FireResponseEvent*. This function is used to invoke an event in the client, but the provision of response data expressed by the UDX model is required by the client. (4) *FireNoResponseEvent*. This function is used to invoke an event in the client that contains the model's calculation results expressed by the UDX model. (5) *FireException*. This function is used to report runtime exceptions to the client.

The model with the *IProcedureModelCore* interface includes *Initialize* and *Running* functions. When the *Running* function is called, the model accepts the input data for calculation. Then, the results are returned as another data form created by the UDX model. In addition to the *Initialize* and *Running* functions, a model with an *ISimulateModelCore* interface also contains a *Kill* function to stop a running process. This model interacts with *IModelSessionStub* in a model execution session.

The purpose of encapsulating the model code is to realize the model stub to achieve interactions between the model and the external environment. There are two cases related to this aspect: modifying the source code and keeping source code intact. If a model's source code can be accessed, the model encapsulation can be realized by modifying the source code, into which a session stub is inserted. If a model's source code cannot be accessed or is unsuitable to modify, a session stub cannot be inserted directly. However, models typically interact with the external environment through input/output (IO) interfaces, whose operation can eventually be converted into Application Programming Interface (API) calls. Therefore, binary interception with hook technology can be adopted to intercept IO calls, in order to transfer IO operations into session stub calls.

4.2 Open cloud computing environment

Model providers register their model as a resource item into the sharing model library, and the model component should be submitted and deployed to a specified model computing node. Such computing node can be selected from our sharing environment, and also, model providers can link their own computing server in order to private their model by themselves. Thus constructing an open sharing and computing environment. Based on the characteristics of virtualization technology in cloud computing field, model execution containers and model message routers were designed to support model execution services (see figure 10).

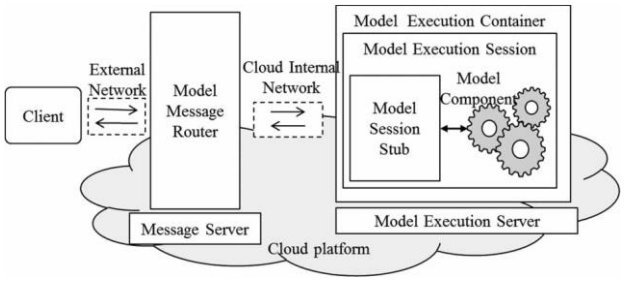


Figure 10 Architecture for the distributed model execution

The architecture to execute geographic models in distributed cloud computing environment consists of two servers: Model Execution Server and Message Server. Model Execution server runs as the model execution container, where the model component be deployed in. And the model execution process would be exposed as model execution session, which translates the geographic model's behavior (request data and response data) as session message. Such session message would not send to the model user (client) directly, the Model Message Server would collect all the model session messages by the cloud internal network, and then route these messages to the corresponding client by external network.

The message, that transmission among model execution server and model message server, embodied the model behavior abstraction (see section 2.3). And the message transfer loop linked all elements in the open sharing and computing platform. The content in such message would "ask" model users to "reply", and the model execution would continue to process according to such "answer".

By meanings of such double level server architecture, model users can get the geographic model executed in distributed cloud computing platform transparently.

5. EXPERIMENT AND DISCUSSION

Based on the strategies mentioned above, a prototype platform was designed. In this platform, VMware which is a kind of virtualization software was firstly installed in blade server branded Inspur™ to construct the cloud environment. Then, six cloud nodes were deployed in this environment; including one agent node, one model resources repository node and four model execution nodes. Based on this, about 500 models, including models in "Resources environment mathematical model manual" and some other geographic analysis models, were expressed by employing unified cloud interface strategy, and then deployed into different execution notes using packaging and deployment strategy. To verify the service capability for collaborative simulation, an improved groundwater process simulation model with more complex interaction was conducted to simulate the changes of groundwater level and recharge in long-time sequence, which required users' parameter input and interactions.

Figure 11 shows the prototype environment that to verify the proposed open architecture. On the left top of figure 11, a simple model represent and edit tool was developed to describe and edit the model's semantic information, behavior information and related data information. The model

encapsulation tool was realized to generate the encapsulation code as showed on the right bottom of figure 11.

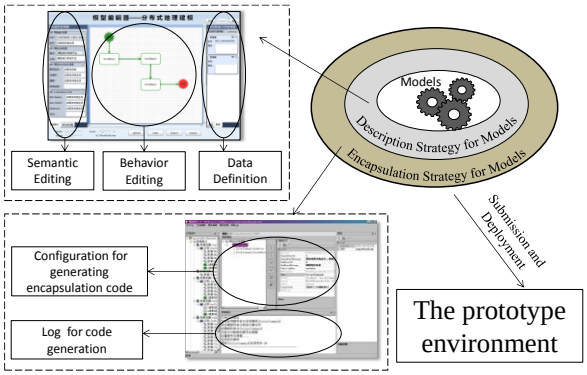


Figure 11 Prototype environment for the open geographic model sharing architecture

Figure 12 shows the results of an improved simulation model for groundwater processes with complex interactions. This model was used to simulate changes in groundwater levels and recharge in a long-time sequence, which required users to input parameters to derive the running process.

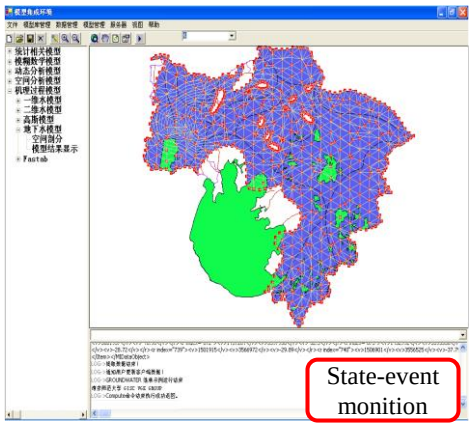


Figure 12 Ground water model execution example

6. CONCLUSIONS AND FUTURE RESEARCH

The key technology of open cloud service strategies for the sharing of heterogeneous geographic analysis models is studied in this article, aiming at solving the problems of sharing heterogeneous geographic analysis models using cloud computing environment. And a relevant experimental environment was established to verify the architecture. The result of the experiment shows that this open architecture was designed with strong adaptability and extendibility to assist the sharing of geographical models.

However, future researches are needed in several aspects: (1) the related tools (e.g. model encapsulation tool, model package and deployment tool and so on) need to be improved so as to make the sharing process more convenient. (2) Due to the limitations of our experimental conditions, our experiments were performed in a simulated virtual environment rather than a real and well-deployed cloud environment. Thus, further research is required on system security and load balancing in the cloud environment.

ACKNOWLEDGEMENT

The authors wish to thank the editor and all the anonymous reviewers. The work described in this paper was supported by the Key Program of the National Natural Science of China Foundation (grant no. 40730527).

REFERENCES

Ahrens, M., 2010. Cloud computing and the impact on enterprise IT. *Lecture Notes in Computer Science*, 63(69), pp. 148-155.

Britz, W., *et al*., 2010. A comparison of CAPRI and SEAMLESS-IF as Integrated Modelling System. Environmental and agricultural modelling, 3, pp. 257-274.

Blind, M., *et al*., 2004. Towards an open modelling interface (OpenMI): The HarmonIT project. In: *Complexity and Integrated Resources Management, Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society*, pp. 346–351.

Crosier, S.J., *et al*., 2003. Developing an infrastructure for sharing environmental models. *Environment and Planning B: Planning and Design*, 30, pp. 487-501.

Diaz, L., *et al*., 2008. An Open service network for geospatial data process. In: *2008 Free and Open Source Software for Geospatial(FOSS4G) Conference*, Cape Town, South Africa, pp. 410-420.

Feng, M., *et al*., 2011. Prototyping an online wetland ecosystem services model using open model sharing standards. *Environmental Modelling & Software*, 26(4), pp. 458-468.

Granell, C., *et al*., 2010. Service-oriented applications for environmental models: Reusable geospatial services. *Environmental Modelling & Software*, 25(2), pp. 182-198.

Hill, C., *et al*., 2004. The architecture of the Earth System Modeling Framework. *Computing in Science & Engineering*, 6(1), pp. 18-28.

Lü, G.N., 2011. Geographic analysis-oriented Virtual Geographic Environment: Framework, structure and functions. Science China (Earth Sciences), 54(5), pp. 733-743.

Leavesley, G. H., *et al*., 2002. A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling. *Hydrological Processes*, 16(2), pp. 173-187

Maxwell, T., and Costanza, R., 1997a. An open geographic modeling environment. *Simulation*, 68(3), 175-185.

Morozov, I., *et al*., 2006. A generalized web service model for geophysical data processing and modeling. *Computer and GeoSciences*, 32 (9), pp. 1403-1410.

Reed, M., *et al*., 1999. A framework for modelling multiple resource management issues-an open modelling approach. *Environmental Modelling & Software*, 14 (6), pp. 503-509.

Valcke, S., *et al*., 2006. PRISM and ENES: a European approach to Earth system modelling. *Concurrency and Computation: Practice and Experience*, 18(2), pp. 247-262.