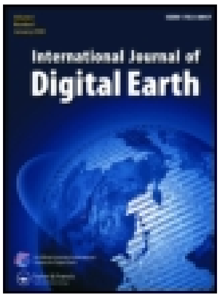


International Journal of Digital Earth
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tjde20

Prototyping an open environment for sharing geographical analysis models on cloud computing platform

Yongning Wen [a] [c] , Min Chen [b] , Guonian Lu [a] , Hui Lin [b] , Li He [a] & Songshan Yue [a]

[a] Key Laboratory of Virtual Geographic Environment (Ministry of Education of China), Nanjing Normal University, Nanjing, China

[b] Institute of Space and Earth Information Science, The Chinese University of Hong Kong, Shatin, Hong Kong

[c] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China

Accepted author version posted online: 01 Aug 2012. Version of record first published: 28 Aug 2012

International Journal of Digital Earth,
2012, 1–27, iFirst article

Prototyping an open environment for sharing geographical analysis models on cloud computing platform

Yongning Wen[a,c], Min Chen[b]*, Guonian Lu[a], Hui Lin[b], Li He[a] and Songshan Yue[a]

[a]*Key Laboratory of Virtual Geographic Environment (Ministry of Education of China), Nanjing Normal University, Nanjing, China;* [b]*Institute of Space and Earth Information Science, The Chinese University of Hong Kong, Shatin, Hong Kong;* [c]*State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China*

(*Received 30 November 2011; final version received 26 July 2012*)

The sharing of geographical analysis models is of crucial importance for simulating geographic processes and phenomena in the current geographical information systems (e.g. Digital Earth), but there remain some issues that have not been completely resolved. The challenges include, eliminating model heterogeneity and searching for suitable infrastructures to support the open sharing and effective execution of models. Taking advantage of cloud computing, this article aims to address the above issues and develop an open environment for geographical analysis model sharing. On the basis of the analysis of the applicability of cloud computing, the architecture of the open environment is proposed. More importantly, key strategies designed for heterogeneous model description, model encapsulating as well as model deploying and transparent accessing in the cloud are discussed in detail to establish such an environment. Finally, the prototype environment is implemented, and experiments were conducted to verify the environment's feasibility to support the sharing of geographical analysis models.

Keywords: geographical analysis models; cloud computing; open environment

1. Introduction

Solving comprehensive geographic problems (e.g. multi-scale problems, regional correlation problems) often requires the reuse or the integration of multi-disciplinary geographical model resources (Maguire *et al.* 2005). In such terms, current geographical information systems (e.g. Digital Earth and the Virtual Geographical Environment) have increased their focus on geographic analysis model base building, which provides a new possibility of model sharing and reuse (Crosier *et al.* 2003, Craglia *et al.* 2008, Yang *et al.* 2008, Lin *et al.* 2009, Lü 2011). The challenge is that although many excellent models have been built, without effective model sharing strategies, these models often work in isolation and cannot communicate with other models and disciplines. Users from different domains have difficulty in sharing their modelling ideas and therefore cannot conduct collaborative work to solve interdisciplinary problems. In addition to the self-characteristics of geographical analysis models, such as heterogeneity (including structural heterogeneity, data heterogeneity and execution environment heterogeneity), dispersion and privacy,

*Corresponding author. Email: chenmin0902@cuhk.edu.hk

ISSN 1753-8947 print/ISSN 1753-8955 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/17538947.2012.716861
http://www.tandfonline.com

other factors, including the lack of an infrastructure with extensive adaptability and relevant service strategies, have greatly affected the prediction capacity of modern geographical simulation (Goodchild *et al.* 1993, Ungerer and Goodchild 2002, Argent 2004, Yang *et al.* 2010). In this case, designing suitable strategies and frameworks for geographic analysis model sharing is undoubtedly becoming one of the inevitable requirements for current geographic research (Nativi *et al.* 2012).

Research on geographical analysis model sharing can be traced back to the development of spatial decision support systems when geographical model bases were first constructed (Muhanna 1993, Bennett 1997, Ascough *et al.* 2002). Although some disciplinary models are successfully integrated into these systems for geographic problem analysis and solving, such as sustainable management of land and water ecosystems (Rizzoli and Young 1997, Oxley *et al.* 2004, Matthies *et al.* 2007), they have generally been run on standalone computers or in closed networks and cannot be shared and integrated for multi-disciplinary simulations (Goodchild 2005).

More recently, different modelling and reuse frameworks have been developed to tackle the environmental problems (Morozov *et al.* 2006, Argent *et al.* 2006), including the Spatial Modelling Environment (SME) (Maxwell and Costanza 1997a, 1997b), the Interactive Component Modelling System (ICMS) (Reed *et al.* 1999), the Dynamic Information Architecture System (DIAS) (Campbell and Hummel 1998, Simunich *et al.* 2002), the Modular Modelling System (MMS) (Leavesley *et al.* 1996, Leavesley *et al.* 2002), ModCom (Hillyer *et al.* 2003), Tarsier (Watson and Rahman 2004), the Community Surface Dynamic Modelling System (CSDMS) (CSDMS Working Group 2004), the Earth System Modelling Framework (ESMF) (Hill *et al.* 2004), the Object Modelling System (OMS) (Ahuja *et al.* 2005), the Open Modelling Interface (OpenMI) (Blind and Gregersen 2005, Moore and Tindall 2005), the European Union's Program for Integrated Earth System Modelling (PRISM) (Valcke *et al.* 2006), the SEAMLESS-IF(Van Ittersum *et al.* 2008) 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 (Nativi *et al.* 2012). On the other hand, few of these frameworks support model sharing and integration over the Internet (Feng *et al.* 2011).

To share model resources as worldwide services, further solutions have been proposed based on Service-oriented architectures (SOA) (Tsou and Buttenfield 2003, Granell *et al.* 2010). It is said that there are at least two advantages in using SOA-based model sharing (Feng *et al.* 2011): one is that SOA can integrate data and models from anywhere in the world (Diaz *et al.* 2008) and the other one is that SOA-based model sharing can apply to various popular applications (Grimshaw *et al.* 2009). To make full use of SOA, the Open Geospatial Consortium and the Technical Committee 211 of the International Standards Organization (ISO/TC211) have established a series of specifications and standards for geographical information sharing Open GIS Consortium (OGC 2004, 2008). In addition, several researchers have investigated model sharing based on Simple Object Access Protocol and Web Services Description Language (SOAP/WSDL) (Fook *et al.* 2009), Web Processing Service (WPS) specifications (Feng *et al.* 2009, Feng *et al.* 2011) and 'Model Web', proposed by Geller and Turner (2007), Geller and Melton (2008) and Geller (2010). However, even with SOA, the efficiency of model resources sharing and the extensibility of model interoperability need to be further improved and related specifications still lack necessary constraints on the model data, the running behaviour and the life cycle.

In the IT domain, the demand for transparent use of computing and software resources has been promoting the development of the IT infrastructure. Cloud computing, which has developed rapidly in recent years, facilitates service-oriented resource sharing in a single large step. Compared with SOA, cloud computing is focused on realising Infrastructure as Service (IaaS) in the infrastructure layer, providing Platform as Service (PaaS) in the application infrastructure layer and user only pays attention to Software as Service (SaaS) in the application layer (Ahrens 2010, Armbrust 2010). All these improved features will create important opportunities for the services of geographical analysis model sharing (Yang *et al.* 2010, 2011).

In this article, we propose an open environment for the sharing of geographical analysis models based on the cloud computing architecture. 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: based on the analysis of the problems encountered in geographical analysis model sharing and the usability of cloud computing architecture, Section 2 proposes the architecture of the open environment for geographical analysis model sharing. In Section 3, a flexible strategy for describing heterogeneous models is introduced. On the basis of an analysis of the model execution service mode in the cloud, an encapsulation strategy for models is described in Section 4 and cloud deployment and service interaction strategies are described in Section 5. Section 6 introduces the realisation of the prototype system and related experiments. Finally, conclusions and discussions of the system's limitations are presented in Section 7.

2. Design of the architecture

2.1. Problems in geographical analysis model sharing

Generally, geographical analysis models can be represented as mathematical models and implemented as executable programmes with specific parameters and runtime platform requirements. Today, the main reasons for the current difficulty in geographical analysis model sharing can be summarised in the following points:

(1) Heterogeneity of model resources: Geographical analysis models vary greatly in modelling method, scale, data organisation and data structure; therefore, it is difficult to adopt fixed description models to represent these geographical analysis models. Moreover, the realisation of these models depends on the diverse hardware architectures, the software platforms and the programming languages, resulting in heterogeneity at the software level (Chen *et al.* 2009a). Due to the heterogeneity of model resources, it is difficult to share them as unified services.
(2) Dispersion of model resources: Although a large number of geographical models have been developed, they are often dispersed across different

institutions or among different researchers. How to collect distributed models for open sharing (OS) is another problem.

(3) Intensive demand for computing resources: Model execution often involves mass geographical data processing and running of complex equations and long-sequence simulations, which requires a large amount of computing resources (NRC 2010). To support effective and distributed execution of the shared model, architecture with powerful computing resources is urgently needed.

2.2. Applicability of cloud computing in supporting the sharing of geographical analysis models

Cloud computing is the integration of several advanced IT and ideas, including supercomputing technology, network communication technology, virtualisation technology and SOA. Simultaneously, cloud computing has also established a clear business model and won support from many enterprises, governments and academic communities (Buyya *et al.* 2008). The current providers of cloud services include Amazon EC2 (http://aws.amazon.com/ec2), Google Docs (http://docs.google.com), Apple iCloud (http://www.apple.com/icloud) and Windows Azure (http://www.microsoft.com/windowsazure). In reference to the five characteristics of cloud computing (The Higher ED CIO 2011), several points can be summarised to indicate that cloud computing can effectively support the construction of an open environment for the sharing of heterogeneous geographical analysis models.

(1) Transparency and openness. 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.
(2) Extensible service. With a huge pool of available computing resources and its extensible architecture, cloud computing is capable of providing sufficient computing resources for the solution of computer-intensive geographical problems.
(3) Adaptability based on virtualisation technology. Virtualisation technology enables cloud computing service providers to offer the required runtime environment dynamically (including specific hardware architecture, operating systems and runtime environments) for heterogeneous geographical model execution.
(4) Alternative services. Because enterprises, scientific research institutions and government agencies have increasingly participated in cloud computing, users may freely and alternatively choose service providers based on their needs in the future.

2.3. Architecture of the open environment

Based on the above analyses, we designed an open environment that supports the sharing of both models and computing resources, making use of cloud computing. Its architecture is presented in Figure 1, including two basic platforms: the Open

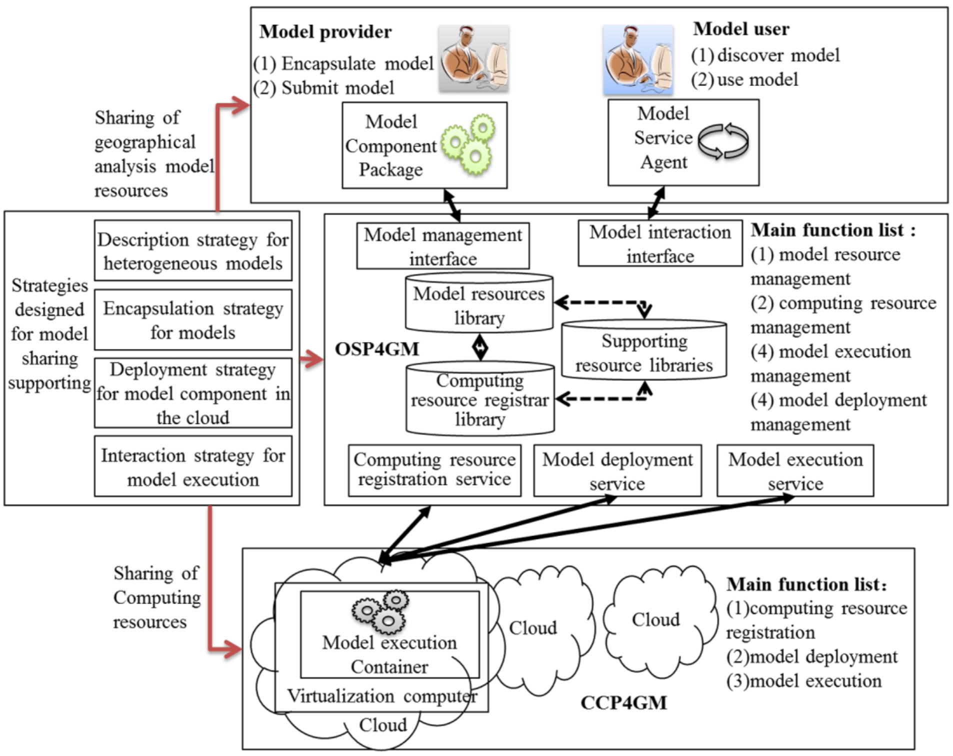

Model provider
(1) Encapsulate model
(2) Submit model
Model user
(1) discover model
(2) use model
Model Component Package
Model Service Agent
Sharing of geographical analysis model resources
Strategies designed for model sharing supporting
Description strategy for heterogeneous models
Encapsulation strategy for models
Deployment strategy for model component in the cloud
Interaction strategy for model execution
Model management interface
Model interaction interface
Main function list :
(1) model resource management
(2) computing resource management
(4) model execution management
(4) model deployment management
Model resources library
Supporting resource libraries
OSP4GM
Computing resource registrar library
Computing resource registration service
Model deployment service
Model execution service
Sharing of Computing resources
Model execution Container
Virtualization computer
Cloud
Cloud
Cloud
CCP4GM
Main function list:
(1)computing resource registration
(2)model deployment
(3)model execution

Figure 1. Architecture of the open environment for the sharing of heterogeneous geographical models based on cloud computing.

Sharing Platform for Geographical Models (OSP4GM) and the Cloud Computing Platform for Geographical Models (CCP4GM).

OSP4GM is the basic platform for model sharing and consists of a set of resource libraries, services and interfaces with the functions of model and computer resource management, model deployment management and model execution management. CCP4GM is responsible for providing computing resources with functions including model execution container creation, model deployment and model execution.

There are also two kinds of actors in this environment. The model providers are responsible for providing the model resources; they can encapsulate heterogeneous models using relevant tools to form standardised model component packages. Moreover, models can be updated or deleted via OSP4GM. After they submit the models, the model users can search and invoke model services via OSP4GM. The requested models are then executed on CCP4GM.

To make full use of the features of cloud computing as well as realise this architecture, several key strategies should be designed. First, to reduce the heterogeneity of models and clearly express the model data and the runtime behaviours, a flexible strategy for the model description is necessary. Second, the strategy for heterogeneous geographical model encapsulation equips models with standardised interfaces; it will contribute to forming model components in the cloud environment and enable modes to be transparently accessed. Third, the cloud deployment and interaction strategy benefits the model execution using virtualisation technology; it will realise the computer resource sharing in the cloud environment. A detailed discussion of the design of these strategies is presented in the next three sections.

3. Description strategy for geographical analysis models

Geographical analysis models are the abstractions of geographical information and geographical processes. The data retrieval and processing steps require knowledge of specific domains. Moreover, when a comprehensive problem is analysed, several geographical models must be integrated and implemented collaboratively. In this case, a flexible description mechanism related to the model data and the execution behaviour will create a foundation for model information exchange and component interface design, further to contribute to model reuse and sharing.

3.1. Data description strategy

Due to the complexity of geographical problems and the diversity of modelling methods, different models have their own data requirements, which involve different contents and can be represented as different data models and formats. It is difficult to design a fixed data model to organise all of the geographical model data (Argent 2004).

To solve this problem, a flexible data description and exchange schema would greatly contribute to the description ability for heterogeneous models (Chen *et al.* 2011). The target of designing such a schema is not to provide a fixed description about the model data requirement or to focus on building a unified library for model contents (e.g. semantic library), but to provide an open description mechanism to help model providers express and exchange model information in a structured and explicit way. In this way, the data structure can be well defined first. Then, by importing of the shared semantic library or the knowledge base, related tags can be attached to the data structure visually and conveniently, and finally, the model-related data can be expressed with exact meanings.

Referenced by former studies in this aspect, which have generally employed XML and its extension language as the basic description language (Adams 2009, Athanasis *et al.* 2009), the UDXDL (Universal Data eXchange Definition Language) is proposed in this article. The difference is that we have designed a data representation model as its core concept. The data representation model was first proposed in the Source for Environmental Data Representation & Interchange technologies (SEDRIS technologies) (http://www.sedris.org/), and later studied primarily in the simulation field (Macchi and Sims 2002, Bhatt *et al.* 2004, Campos and Hull 2004, Bhatt *et al.* 2005), rather than the geographic modelling field (Chen *et al.* 2009b). In this case, we designed our data representation model for geographic model data description, which titled the UDX model written in XML to form UDXDL. The UDX model consists of two parts: the UDX object model, which is used to express the data structure and the semantically enhanced schema, which is used to attach UDX objects with semantics. With the UDX model, geographical model data containing semantics and constraints can be expressed through combinations and iterations of different basic elements of the two parts.

3.1.1. UDX object model

Generally, geographical data are organised based on several rules. (1) Simple variables and variable sets are the most common basic elements. (2) A hierarchical structure and a two-dimensional table structure are the most general forms of data organisation. Moreover, the relational model, which is essentially organised based on a two-dimensional table, has been demonstrated to enable the expression of complex data relationships (Rafanelli 2003). (3) Naming is the simplest way to distinguish data contents. The names of data items, data tables and data structures contain certain intelligible information. (4) The meaning of a data structure generally relies on its explanation. For example, a point sequence can be viewed as a LineString or a Ring. With additional triangle information, a point sequence can also represent the nodes in a Triangulated Irregular Network (TIN). Based on the above analysis, a UDX object model was designed. Its Unified Modeling Language (UML) class diagram is shown in Figure 2.

The UDX object model includes two types of basic construction elements: the nodes and the kernel. Each node has a name, a kernel, and 0 to *n* child nodes. The kernel dictates each node's behaviour, its contained data and its type. The kernel subclasses include the value kernel, the array kernel and the container behaviour-type kernel.

The template types for the value kernel and the array kernel are entitled *KnValueT* and *KnArrayT*. *KnValueT* is used to store simple value data types, whereas *KnArrayT* stores arrays of simple value data types. The simple value types that can be stored in *KnValueT* include *Int*, *Float*, *String* and *Vector* (such as *Vector2D*, *Vector3D* and *Vector4D*, which are designed specifically for spatial data). When the kernel of a specific node is a value kernel or an array kernel, this node cannot have any child nodes.

When a node contains the container behaviour-type kernel, it does not store data; instead, the enumeration types of a container behaviour-type kernel indicate the types of its child nodes, which include *Structure*, *List*, *KeyValue* and *Table*, and can be described as follows. (1) *Structure*: The kernel-subordinated node is structurally similar to the type of the programming language; this type of node may contain child nodes with any kernel. (2) *List*: The child nodes of the kernel-subordinated node have the same node hierarchy. (3) *KeyValue*: The kernel-subordinated node has two array-type child nodes of the same length, and the two corresponding array elements form

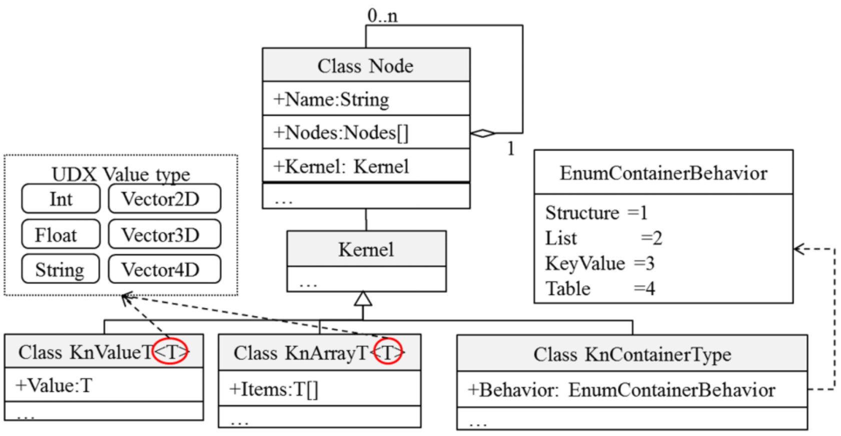


Figure 2. UML class diagram of the UDX object model.

key-value mappings. The first array element indicates the keys and the second indicates the values. (4) *Table*: The kernel-subordinated node has several child nodes, which are array-type nodes with the same number of elements. All child nodes constitute a two-dimensional table with the elements at the same positions forming the rows of the table.

Examples of a data representation using the UDX object model are shown in Figure 3. Figure 3(a) shows a TIN. The *Points* child node indicates that the *Vector2D* array is used to express all of the point sequences; the *Triangles* child node indicates that the sequence of triangles is composed of points, and the array elements are used to represent the index of vertex. Figure 3(b) demonstrates a polygon. The *Patches* child node indicates the ring structures of polygons, and the value of each array element is used to represent the number of points contained in each ring. Figure 3(c) shows a grid Digital Elevation Model (DEM). *XCount* and *YCount* represent rows and columns of the DEM, and *ZValues* represents the elevation points. Figure 3(d) shows an expansion of the grid DEM structure with additional information concerning the DEM's *Origin* and *Span*.

3.1.2. Semantically enhanced schema

Semantic information is used to give meaning to the data. Conveying semantic information in an explicit way will greatly facilitate model sharing and assist in intelligent parameter matching, model connection and data validation. However, this information is difficult to express directly using data structures and parameters in data files. As an example, Figure 4 is a representation of data for a water flow problem; input data are shown on the left, and their meanings are illustrated graphically in the right hand side sub-figure.

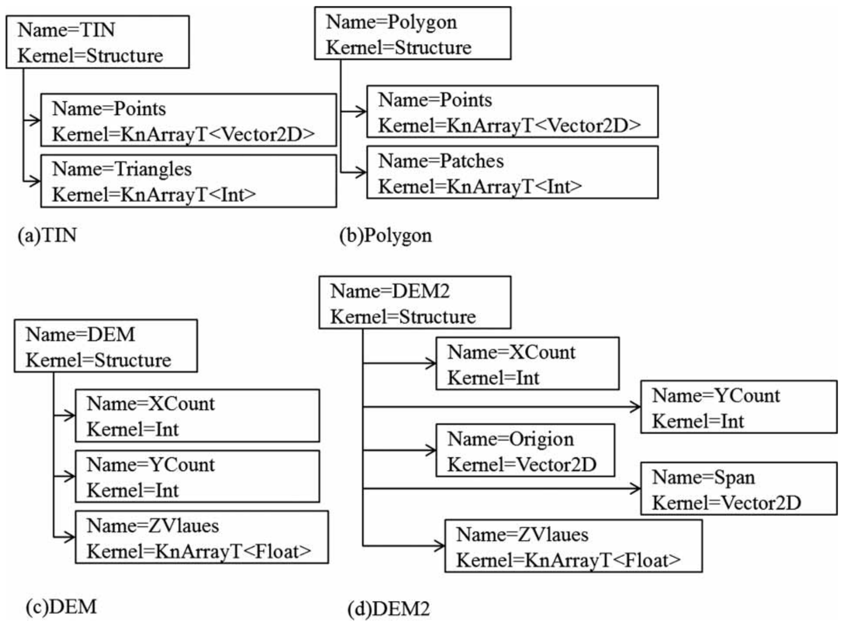


Figure 3. Examples of data representation using the UDX object model.

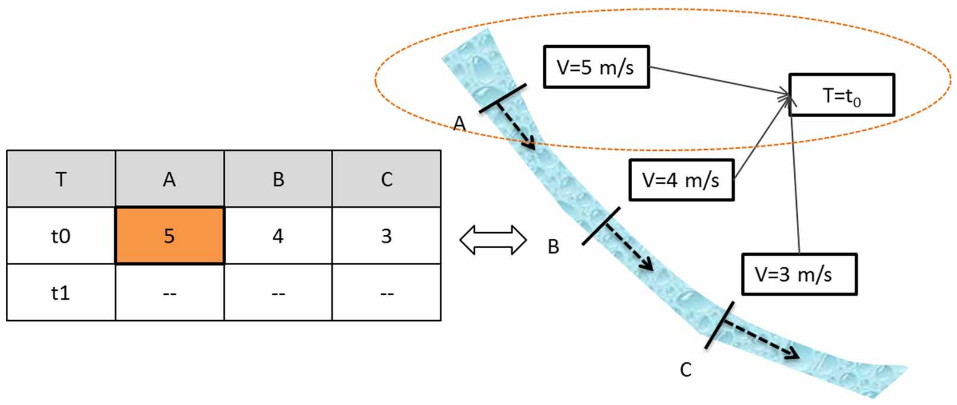
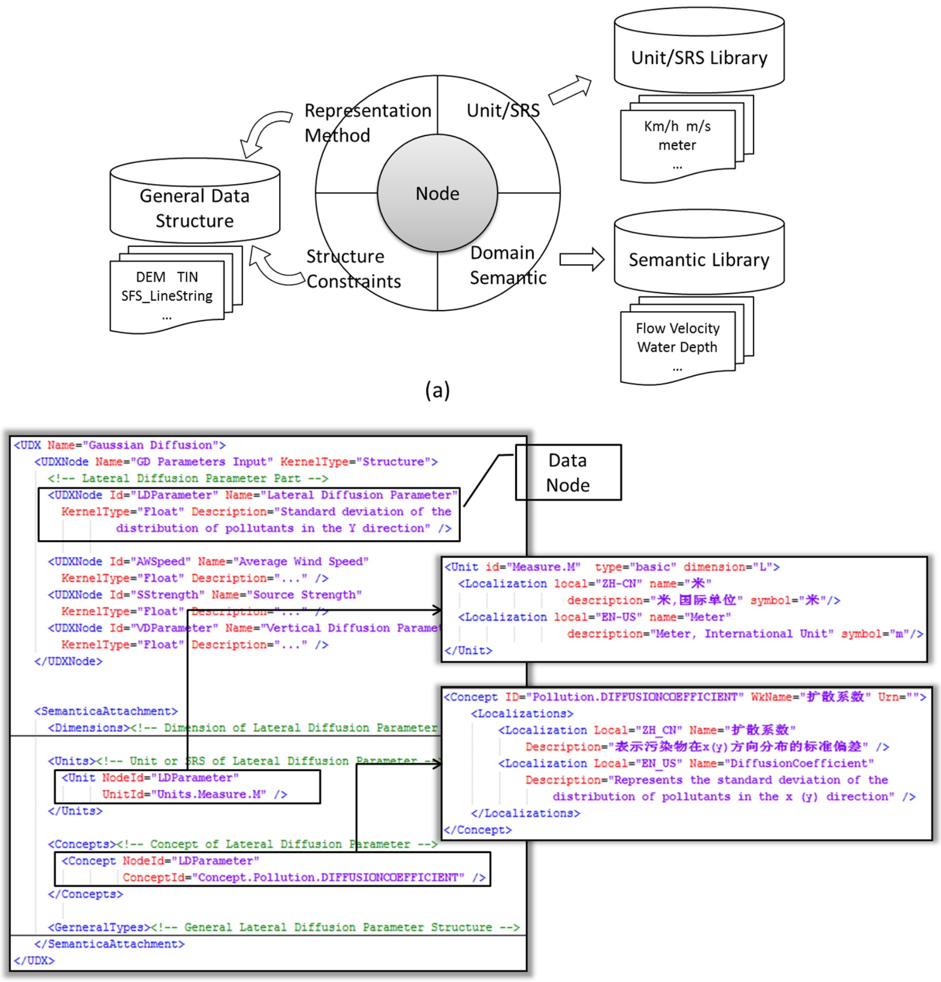


Figure 4. Semantics explanation of data.

For example, data item '5' indicates extensive information, such as (1) the unit of flow velocity is m/s rather than km/h, (2) the input data are the flow velocities rather than the pollutant concentrations or the temperatures and (3) the data describe the flow velocity for water flowing through cross-section A at time t0.

Although the UDX object model can be used to express complex data structures and contents, it cannot express the semantic information implied in the data. To address this shortcoming, a flexible semantic supplement strategy is important for geographic data description. Since the 1990s, with the aim of reducing the cognitive differences among geographic experts of multiple domains, related research on geographical ontology and cognition was conducted and applied to the semantic description of geographic data; to date, many achievements have been recorded (Kuhn 1994, Mark *et al.* 1999, Mennis *et al.* 2000, Kuhn 2001, 2002, 2003, 2005, Visser *et al.* 2002, Galton 2003, Smart *et al.* 2004, Abdelmoty *et al.* 2005). However, due to the complexity of geographic data and the difference of geographic cognition, the study of semantic abstract and expression of geographic data is still in its infancy and the problem of semantic heterogeneity has yet to be completely resolved (Donato 2010).

Referenced by SEDRIS, this article does not focus on the study of the semantic or geographic ontology, but provides a mechanism to clearly express the semantic so that both users and providers can share the semantic explicitly to help reduce the heterogeneity of the data. To establish a solid foundation for the sharing of geographic model data, the semantically enhanced schema was realised using XML, with four parts considered as follows: (1) measurement units and a spatial reference system (SRS); (2) meaning of data, such as soil porosity and flow velocity; (3) general data representation methods in specific domains (e.g. a TIN structure is generally defined with node sets and triangle sets); and (4) constraints between data structures (e.g. the number of triangle vertex indexes in a TIN must be no greater than the number of points in the point sequences). Figure 5(a) shows the design of the schema, and Figure 5(b) shows an example that can be used to illustrate the data used for Gaussian diffusion with semantic tags. In this case, the exact contents of the semantic can be imported by some mature semantic libraries according to the different classification; semantic tags can be attached to the UDX object model using Key-Value mode. For example, ArcGIS (http://www.esri.com/software/arcgis/index.html) or SEDRIS's (http://www.sedris.org/) SRS can be imported as the spatial



```
<UDX Name="Gaussian Diffusion">
  <UDXNode Name="GD Parameters Input" KernelType="Structure">
    <!-- Lateral Diffusion Parameter Part -->
    <UDXNode Id="LDParameter" Name="Lateral Diffusion Parameter"
      KernelType="Float" Description="Standard deviation of the
              distribution of pollutants in the Y direction" />

    <UDXNode Id="AWSpeed" Name="Average Wind Speed"
      KernelType="Float" Description="..." />
    <UDXNode Id="SStrength" Name="Source Strength"
      KernelType="Float" Description="..." />
    <UDXNode Id="VDParameter" Name="Vertical Diffusion Parame
      KernelType="Float" Description="..." />
  </UDXNode>


  <SemanticaAttachment>
    <Dimensions><!-- Dimension of Lateral Diffusion Parameter

    <Units><!-- Unit or SRS of Lateral Diffusion Parameter -
     <Unit NodeId="LDParameter"
           UnitId="Units.Measure.M" />
    </Units>

    <Concepts><!-- Concept of Lateral Diffusion Parameter -->
     <Concept NodeId="LDParameter"
              ConceptId="Concept.Pollution.DIFFUSIONCOEFFICIENT" />
    </Concepts>

    <GerneralTypes><!-- General Lateral Diffusion Parameter Structure -->
  </SemanticaAttachment>
</UDX>
```

```
<Unit id="Measure.M"  type="basic" dimension="L">
  <Localization local="ZH-CN" name="米"
                  description="米,国际单位" symbol="米"/>
  <Localization local="EN-US" name="Meter"
                  description="Meter, International Unit" symbol="m"/>
</Unit>
```

```
<Concept ID="Pollution.DIFFUSIONCOEFFICIENT" WkName="扩散系数" Urn="">
    <Localizations>
        <Localization Local="ZH_CN" Name="扩散系数"
            Description="表示污染物在x(y)方向分布的标准偏差" />
        <Localization Local="EN_US" Name="DiffusionCoefficient"
            Description="Represents the standard deviation of the
                distribution of pollutants in the x (y) direction" />
    </Localizations>
</Concept>
```

Figure 5. Semantically enhanced schema for UDX.

reference tags of the data, while the Unit system of Ptolemy (http://ptolemy.eecs.berkeley.edu/) can be imported for unit tags.

By embedding the semantic information into the UDX object model, the UDX model can be formed and employed to represent both data values and semantics. In this way, various types of semantic information can be attached to nodes with different types of kernels. For example, Unit/SRS and semantic information can be attached to nodes with value-type kernels, whereas data structure template information can be attached to nodes with structure-type kernels.

3.2. *Behaviour description strategy*

3.2.1. Analysis of model execution behaviour

Although the internal structures of geographical analysis models are generally complicated, their external interfaces are relatively simple. In the present case, with respect to the users, the geographical analysis model can be regarded as a black box, and the model's behaviour can thus be considered as interactions between the model and the users. Therefore, a strategy for the description of the model behaviour was developed according to two interaction types: simple interaction behaviour and complex interaction behaviour.

The description of the simple interaction behaviour is used to express the execution behaviour of the model with only one data processing flow or calculation process. The single process may include data submission by the user, model calculation, or the presentation of results. The model does not interact with other models or users throughout the process.

3.2.2. *Description of the complex interaction behaviour*

The complex interaction behaviour, which is usually used in simulating complex geographical processes, refers to execution processes that require multiple interactions with the external environment. These interactions include file reading and writing, requests user interventions and interoperation with other models.

To describe such complex interaction processes, a state machine-event response (SM-ER) model has been designed (see Figure 6). Because the SM is used in this model, whereas user interactions are represented by responses to different events, the model is defined as an SM-ER model.

The concept of the SM-ER model is shown in Figure 6(a). After the model is launched, the SM-ER model begins to change state with its own simulation logic and triggers different events (e.g. requesting or providing data) until the entire simulation is completed. In this case, the whole process can be regarded as transitions from one state to another in which the interaction is realised by sending interaction events and waiting for external responses.

Figure 6(b) shows the UML model of the SM-ER model. 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

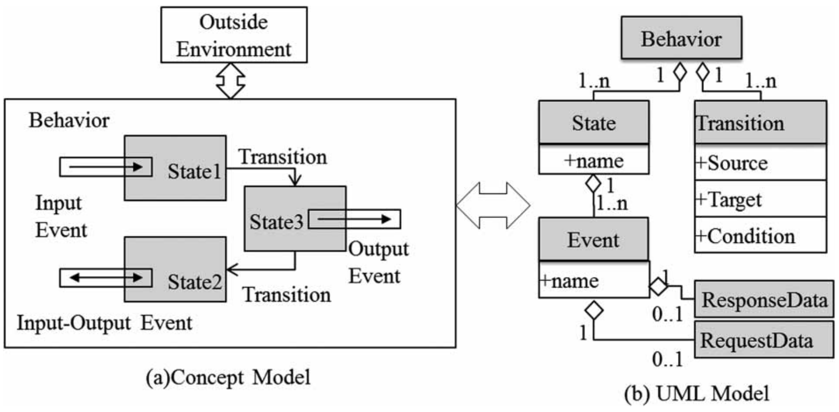


Figure 6. SM-ER model designed for describing complex interaction behaviour.

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.

Because the SM-ER model can represent either a strict execution order or a data requirement, combined with the UDX model, it can clearly reflect the execution process and benefit collaborative interactions between the models and the external environment.

4. Model encapsulation strategy based on cloud computing

According to the architecture proposed in Figure 1, the description strategy for the model resources designed in the previous section helps to reduce the heterogeneity in the model itself. However, model execution usually relies on a particular platform. To address the problem of platform heterogeneity, it was necessary to design a related encapsulation strategy to enable transparent access to models in the cloud.

4.1. Model execution service based on virtualisation technology

Virtualisation technology establishes a critical foundation for the realisation of cloud computing. Using this technology, different hardware platforms can be virtualised by the same hardware environment and can further be used for deploying a compatible OS and installing corresponding software environments (Siddhisena *et al.* 2011). Based on the characteristics of virtualisation technology, model execution containers and model message routers were designed to support model execution services (see Figure 7).

A model execution container serves as the host for a model that is running on a virtual platform. To isolate different models, the model execution container creates separate sessions for each service and creates session stubs for models interacting with the external environment. The model message routers run on independent servers in CCP4GM and are responsible for forwarding messages between the model execution containers via a high-speed communication mechanism in a cloud computing environment.

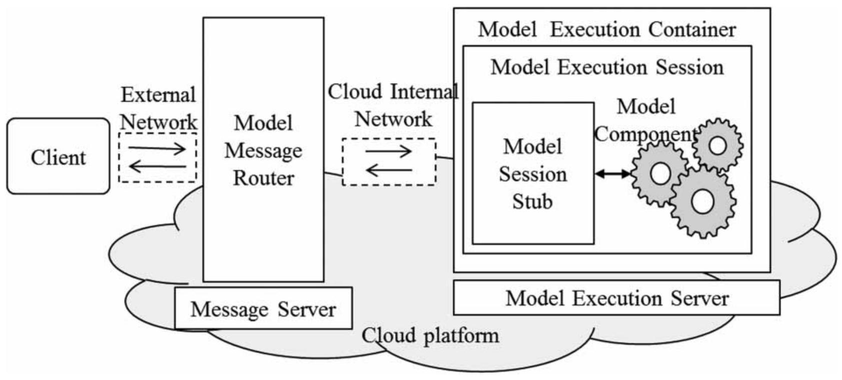


Figure 7. Model execution service based on virtualisation technology.

4.2. Model encapsulation strategy

4.2.1. Model component interface design

The model runs in a sandbox environment provided by a model execution container. Therefore, the interfaces between the models and the containers must follow a set of standards. A model created according to these interface standards is called a standardised model component (Larson *et al.* 2004). Figure 8 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 standardised model components based on the definitions of these interfaces.

An object with an *IModelSessionStub* interface communicates with the external environment following the SM-ER model (see Section 3.2.2). 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 initialisation 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 *Initialise* and *Run* 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 process is shown in Figure 9.

(1) A model execution session calls the *Initialise* function to initialise the model; (2) the *Run* function is called to launch the model; (3) the model invokes the *SwitchState* function to notify the client to enter *State0*; (4) the model triggers the *GetParam*

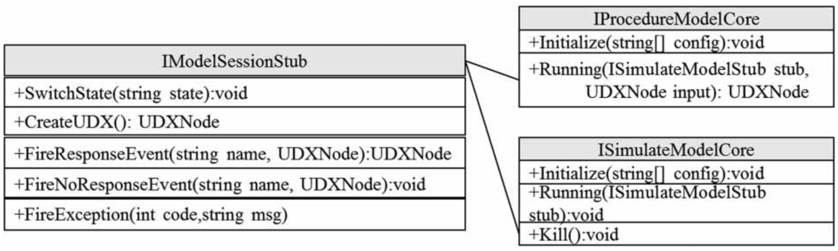

IModelSessionStub
+SwitchState(string state):void
+CreateUDX(): UDXNode
+FireResponseEvent(string name, UDXNode):UDXNode
+FireNoResponseEvent(string name, UDXNode):void
+FireException(int code,string msg)
IProcedureModelCore
+Initialize(string[] config):void
+Running(ISimulateModelStub stub, UDXNode input): UDXNode
ISimulateModelCore
+Initialize(string[] config):void
+Running(ISimulateModelStub stub):void
+Kill():void

Figure 8. The abstract model of the interface standards for the model components.

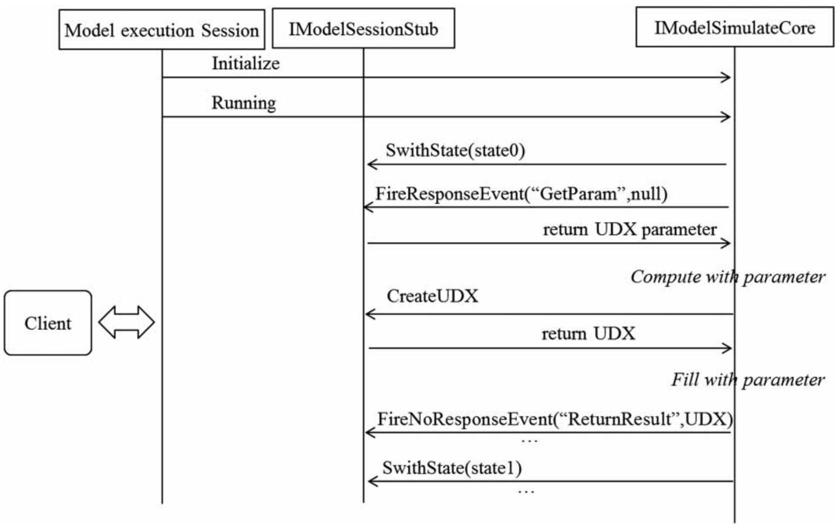


Figure 9. The process of model interaction with the outside environment.

event and requests data from the client; (5) the model begins the calculations; (6) the model requests the *IModelSessionStub* to create data objects expressed by the UDX model; (7) the model fills the data objects based on the calculation results; and (8) a *ReturnResult* event is triggered to return the calculation results. The steps are repeated until model execution is completed.

4.2.2. Heterogeneous model encapsulation strategy

Heterogeneous model encapsulation involves encapsulating the original heterogeneous model code into model components that can be hosted in a model execution container. Two issues arise in this process: whether the model code must be modified and whether the model runs in a host process as an independent process.

(1) The purpose of encapsulating the model code is to realise 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 (see Figure 10).

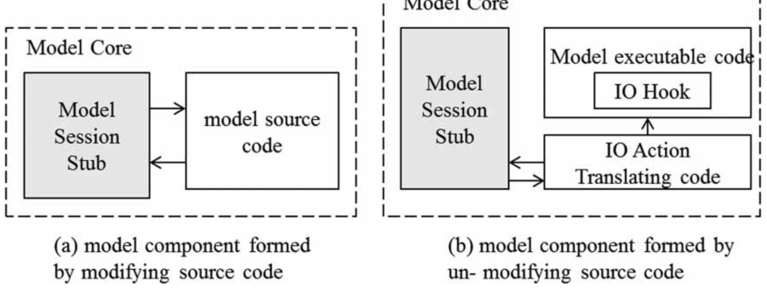

Model Core
Model Session Stub
model source code
(a) model component formed by modifying source code
Model Core
Model Session Stub
Model executable code
IO Hook
IO Action Translating code
(b) model component formed by un- modifying source code

Figure 10. Two modes for model encapsulation by modifying model code.

If a model's source code can be accessed, the model encapsulation can be realised by modifying the source code, into which a session stub is inserted. In this case, a session stub interface should be realised using the same programming language as the model's source code. After the source code is modified, the model components can be obtained by re-compiling and re-linking.

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 (Hunt and Brubacher 1999) can be adopted to intercept IO calls, in order to transfer IO operations into session stub calls.

(2) The executable programme of model components is in the form of either library files or executable files. The former runs in the host process, whereas the latter can only run as an independent process. In this case, objects with *IProcedureModelCore* or *ISimulateModelCore* interfaces are designed differently according to the different running modes (see Figure 11). The core object designed for the latter is only a proxy for the model code and interacts with the real model through a communication mechanism (e.g. inter-process communication channels).

Based on the above analysis, the selection of an encapsulation method is essentially related to the form of the model's code. The model can be easily modified if the source code is available and can also be easily encapsulated by the execution of either an in-process mode or an out-of-process mode. Without the model's source code, the execution of an in-process mode can be used for encapsulation in the case of library files; however, for executable files, only execution in an out-of-process mode is suitable. Generally, execution in an out-of-process mode provides better stability and safety, whereas an in-process mode consumes fewer system resources and can be launched quickly.

5. Cloud deployment and service interaction strategies

5.1. Description of deployment configuration

After models are encapsulated into model components, model components must be packaged and deployed in the cloud for transparent access. Figure 12 shows an example of a deployment package.

The package includes the model components and the model deployment configuration documentation, which is designed to describe information related to the model deployment, access and execution. It includes the following items:

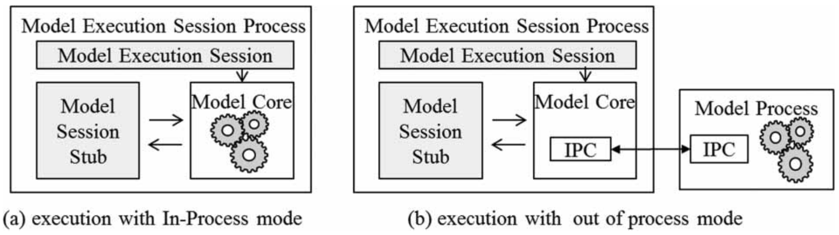
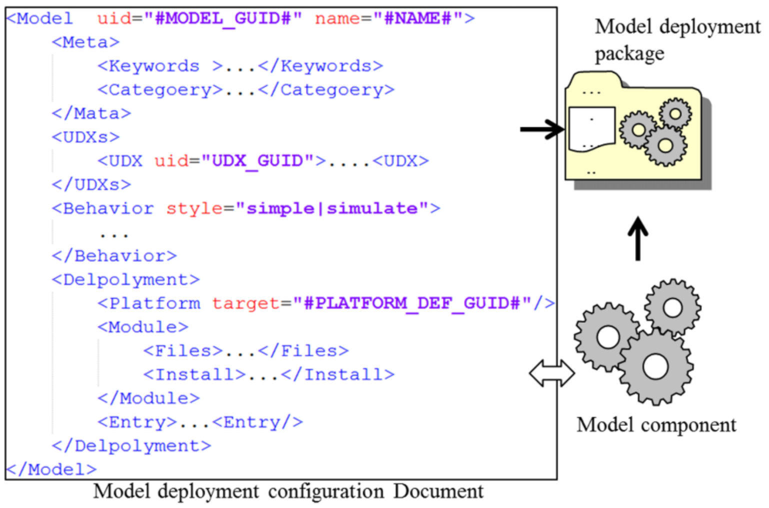

Model Execution Session Process
Model Execution Session
Model Session Stub
Model Core
(a) execution with In-Process mode
Model Execution Session Process
Model Execution Session
Model Session Stub
Model Core
IPC
Model Process
IPC
(b) execution with out of process mode

Figure 11. Two process modes of model execution.



```
<Model  uid="#MODEL_GUID#" name="#NAME#">
    <Meta>
        <Keywords >...</Keywords>
        <Categoery>...</Categoery>
    </Mata>
    <UDXs>
        <UDX uid="UDX_GUID">....<UDX>
    </UDXs>
    <Behavior style="simple|simulate">
        ...
    </Behavior>
    <Delpolyment>
        <Platform target="#PLATFORM_DEF_GUID#"/>
        <Module>
            <Files>...</Files>
            <Install>...</Install>
        </Module>
        <Entry>...<Entry/>
    </Delpolyment>
</Model>
```

Figure 12. Model deployment package and its composition.

(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 UDXSL; (4) a Behaviour node to define the executable behaviour of the model; and (5) a Deployment node to describe model deployment information. This type of node includes the following child nodes: (1) a Platform node to define the executing environment of the model; (2) a Module node to describe the module information for the model deployment; and (3) an Entry node to describe the entry information to launch the model code.

5.2. Registration and deployment strategy of computing and model resources

After the model deployment packages are built, the model and computing resources are bound by the collaboration of OSP4GM and CCP4GM. Finally, executable model services can be provided. OSP4GM is responsible for model and computing resource management as well as deployment coordination along with the core functions of modelling resource registration, computing resource registration, model deployment and service creation. CCP4GM is responsible for physical deployment and execution along with the core functions of model deployment management and model service creation (see Figure 13).

The collaboration process between OSP4GM and CCP4GM is discussed here:

(1) Model submission. Model providers submit model deployment packages through a model resource management interface. OSP4GM is responsible for the integrity verification of the deployment package, an analysis of the configuration information, the creation of a model index and the registration of the metadata. OSP4GM is also responsible for saving the packages in a model resource library for subsequent deployment.

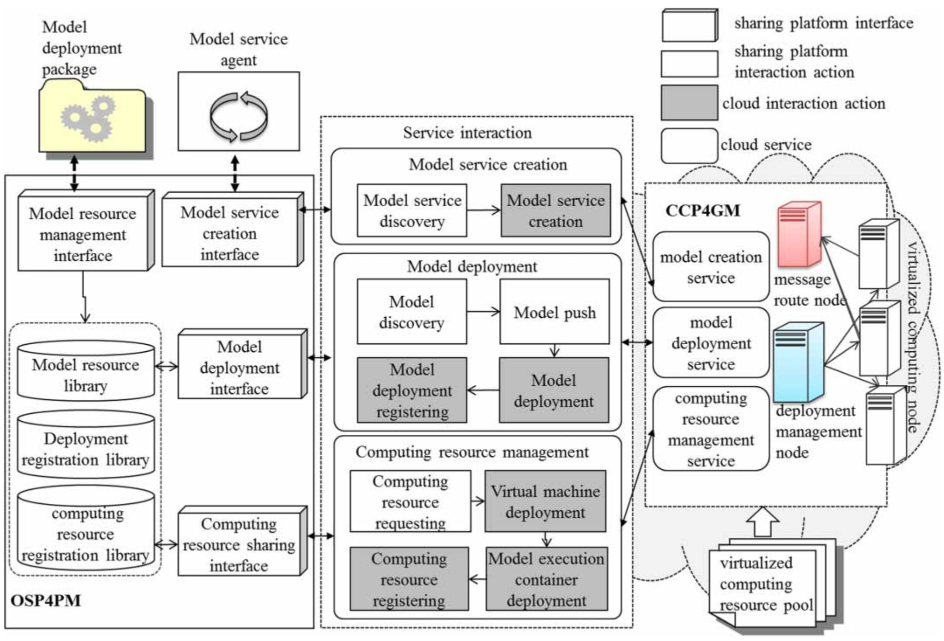

Model deployment package
Model service agent
sharing platform interface
sharing platform interaction action
cloud interaction action
cloud service
Service interaction
Model service creation
Model service discovery
Model service creation
Model deployment
Model discovery
Model push
Model deployment registering
Model deployment
Computing resource management
Computing resource requesting
Virtual machine deployment
Computing resource registering
Model execution container deployment
Model resource management interface
Model service creation interface
Model resource library
Model deployment interface
Deployment registration library
computing resource registration library
Computing resource sharing interface
OSP4PM
CCP4GM
model creation service
model deployment service
computing resource management service
message route node
deployment management node
virtualized computing node
virtualized computing resource pool

Figure 13. Registration and deployment of model and computing resources.

(2) Computing resource management. CCP4GM provides computing resources based on virtualisation technology. A deployment management node and several virtual computing nodes constitute a dynamic virtual computing resource pool. The deployment management node is responsible for creating virtual computing nodes, deploying model execution containers, adjusting computing resources and dynamically updating the registration information of virtual computing resources. This node also maintains a list of computing resources and registers them with OSP4GM. OSP4GM then dynamically requests for the allocation of computing resources based on the registered information. As described in Section 4.1, each computing node in CCP4GM is an independent virtual computer that provides an execution container for a heterogeneous model.

(3) Model deployment. Model components are deployed to the corresponding model execution containers by a model deployment engine according to the description in the model deployment package. Four steps are involved in this process: computing resource discovery, model resource push, model component deployment and deployment information registration. The deployment information is recorded into the model and the computing resource binding list in OSP4GM.

(4) Model service creation and transparent resource deployment. The model service creation engine is responsible for searching for model services, matching them with execution containers and creating model-running cases. OSP4GM-based model service creation is a transparent process including computing resource allocation, model deployment and model service creation, as shown in Figure 14.

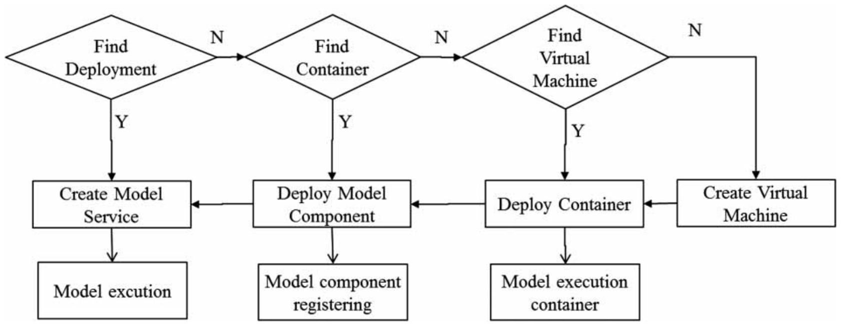


Figure 14. Transparent creation of model service.

During the entire process, deployment registration information is queried in OSP4GM. If there exists a model execution container already hosting a model in the CCP4GM cloud, OSP4GM directly requests the corresponding instances; otherwise, the model deployment process is launched. If no corresponding model execution container is found during the model deployment process, a model execution container is created based on the model deployment configuration files. To improve the response speed for the model service creation, a pre-deployment mechanism is adopted.

6. Experiment

6.1. Experiment environment

We established a simulative experiment environment based on an Inspur^{TM} NX7140 cluster, which contains 28 blades (with 256 **cores**, 1 **Terabyte (TB) of Random Access Memory (RAM)** and 256 TB of hard disk space) and a VMW are ESX Server, which is a virtualisation programme that was installed to simulate an experimental environment consisting of 11 virtual computers (see Figure 15).

In this environment, a virtual computer called the *mds_portal* was used to deploy the OSP4GM server, with the CCP4GM functionality consisting of two components: *mds_c1* and *mds_c2*. As an example, *mds_c1* contains a management node named

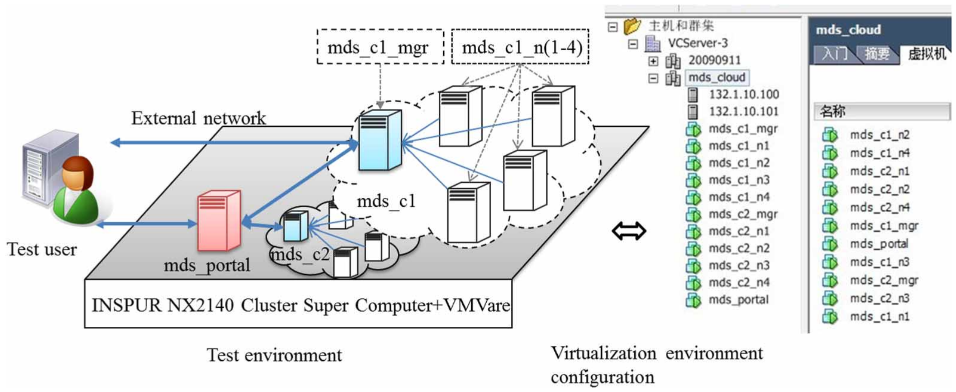


Figure 15. Configuration of experimental environment.

mds_c1_mgr and four other computing nodes, *mds_c1_n1* to *mds_c1_n4*. In addition to managing these internal nodes, the *mds_c1_mgr* node communicates with the *mds_portal* through an external network. Model deployment and message routing services in the cloud were provided by the *mds_c1_mgr* node; the model execution containers were deployed at the computing nodes (*mds_c1_n1* to *mds_c1_n4*).

6.2. The construction of geographical model sharing environment

On the basis of the framework described in Section 2.3, the OSP4GM and CCP4GM platforms were realised. OSP4GM contains several different interfaces and website user interfaces, which are shown in Figure 16. With reference to the Volunteer mode (Goodchild 2009, 2010), each user is not only able to contribute descriptive information resources, including units, semantics and abstract structure UDX model through services provided by the OSP4GM portal but is also able to submit, search and use model component resources.

Model execution containers, a message router and a computing resource deployment management server were realised for the CCP4GM. The model execution container was designed to support the encapsulation of two types of modes, which were implemented in.NET or C/C++. The message router implementation is based on Microsoft IIS 7.0/ASP.NET and can interact with an external environment solely via the HTTP protocol. The computing resource deployment management server works with the model execution container to deploy models from the OSP4GM.

In addition to the construction of basic platforms, experimental resource libraries, including a unit library, a semantic library and a data structure library, were established to help users more clearly describe the models they contributed or required. The unit library was based on the GB-3100-3102/ISO-1000:1992 standard, which includes more than 400 standard units. The semantic library was based on the Dictionary of Modern Geography (Zuo 2009) and was also integrated with the NASA Semantic Web for Earth and Environmental Terminology (SWEET) ontology

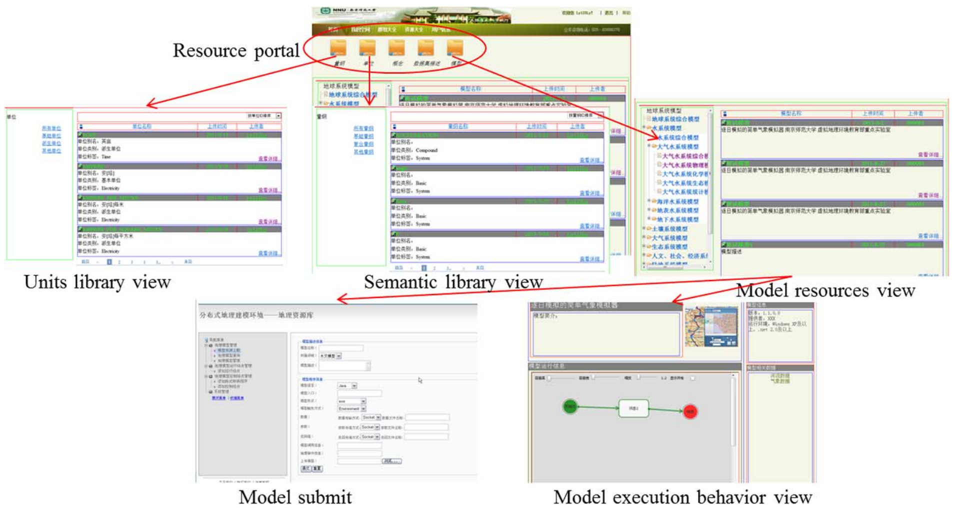


Figure 16. Main interfaces of model sharing environment.

library (Raskin and Pan 2005). Moreover, a relatively comprehensive general semantic library in the field of geographical modelling and simulation was constructed through association with Wikipedia entries. Furthermore, a model classification system was constructed based on the Scientific Classification System of the Earth (Chen 1998) for model indexing. All of these resources can be added by users if they cannot be found in the environment.

6.3. Practical verification of the geographical analysis model sharing environment

To verify the practicability of the environment, the project group organised a testing team with two types of volunteers: a team of model collectors and providers and a team of model users. The former was responsible for model collection, encapsulation, registration and submission using the OSP4GM, whereas the latter was responsible for geographical process simulation using the models deployed in the cloud.

A representative experimental model library was constructed from the contributions of the model collector and provider team. By referring to the Resources and Environment Models Manual (Yue 2003) and 62 analytic models in the fields of atmospheric science, hydrology, meteorology and others, the team realised or reconstructed most of the models and created more than 2000 model components, which were later encapsulated using the strategies previously described in Section 4.

In terms of the implementation process, with the objective of encouraging and helping model owners share their models, a corresponding model description tool and encapsulation tool were developed based on the well-designed strategies of model description and heterogeneous model encapsulation (see Figure 17). The

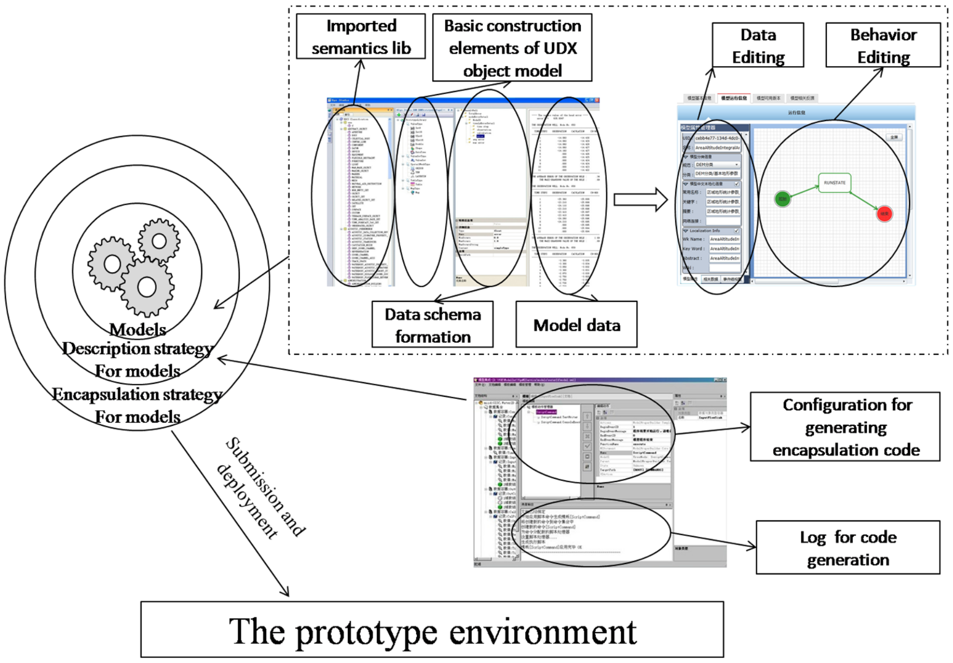

Imported semantics lib
Basic construction elements of UDX object model
Data Editing
Behavior Editing
Data schema formation
Model data
Models
Description strategy For models
Encapsulation strategy For models
Configuration for generating encapsulation code
Log for code generation
Submission and deployment
The prototype environment

Figure 17. Strategies realisation and accordingly tools.

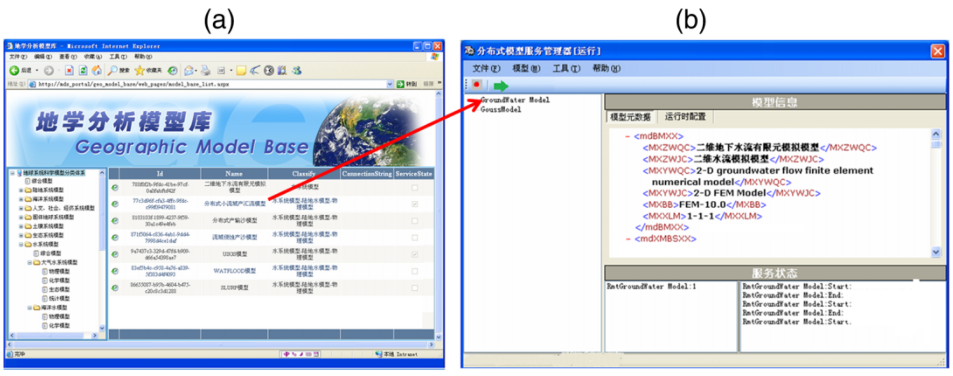
model description tool enables interactive definition of the UDX data description and running behaviour of the models. Tool (a) is responsible for data structure defining and then attaching related semantics tags to data by dragging these tags into data elements in the formed date structure. Tool b takes charge of data editing as well as model behaviour editing. Tool c is the model encapsulation tool. It can automatically generate the conversion code between UDX and the data type of the models that use a text file as the IO, and it can also generate the interaction code between the models and the IModelStub interface based on the model behaviour. However, improvements to the encapsulation tool are needed because it does not support the encapsulation process of models with other forms of inputs and outputs, and requires the manual interaction of the users.

After the models were encapsulated, they were uploaded to the OSP4GM with their descriptions, as described in Section 3, and then deployed in the CCP4GM. Figure 18 shows the deployment of several models. Figure 18(a) shows the model lists and Figure 18(b) shows the configuration information as well as the model execution container of the model component.

Based on the models provided above, the user team searched and transparently used the deployed models with the computing resources provided by *mds_c1* and *mds_c2* in the CCP4GM. Figure 19 shows the results of invoking the model service in the cloud. The results in Figure 19(a) are from a geographical simulation of a Gaussian dispersion model, which was used to simulate the diffusion and concentration distribution of a point source of atmospheric pollution. Figure 19(b) 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.

7. Conclusions and future research

The purpose of this study is to build an open environment for geographical analysis model sharing based on cloud computing environment and to construct a volunteer-style sharing mode for modelling and computing resources. Our main focus was on



```
- <mdBMXX>
    <MXZWQC>二维地下水流有限元模拟模型</MXZWQC>
    <MXZWJC>二维水流模拟模型</MXZWJC>
    <MXYWQC>2-D groundwater flow finite element
      numerical model</MXYWQC>
    <MXYWJC>2-D FEM Model</MXYWJC>
    <MXBB>FEM-10.0</MXBB>
    <MXXLM>1-1-1</MXXLM>
  </mdBMXX>
- <mdXMBSXX>
```

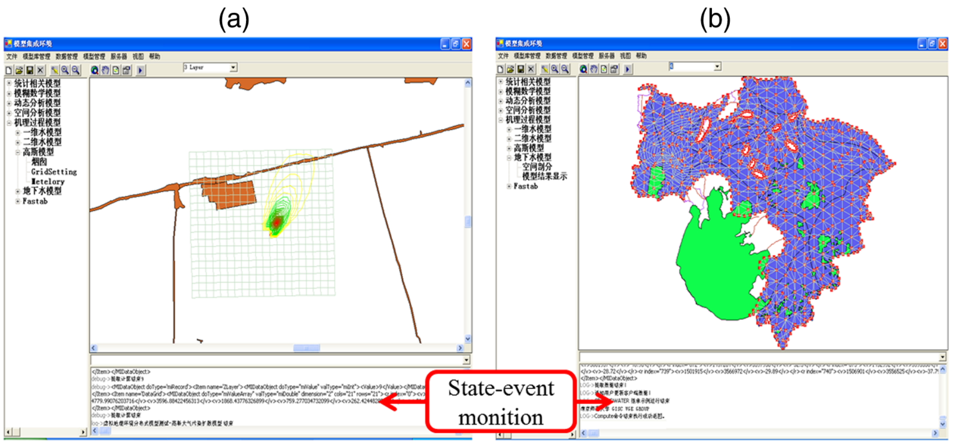
```
RmtGroundWater Model:1
```

```
RmtGroundWater Model:Start:
RmtGroundWater Model:End:
RmtGroundWater Model:Start:
RmtGroundWater Model:End:
RmtGroundWater Model:Start.
```

Model resource list

Model deployment in execution container

Figure 18. Model resource library and model deployment.



```
</Item></MIDataObject>
debug->提取计算结果
debug-><MIDataObject doType="miRecord"><Item name="ZLayer"><MIDataObject doType="miValue" valType="miInt"><Value>9</Value></MIData
</Item><Item name="DataGrid"><MIDataObject doType="miValueArray" valType="miDouble" dimension="2" cols="21" rows="21"><r index="0"><v>
4779.99076203716</v><v>3596.88422456313</v><v>1868.43776326899</v><v>759.277034732099</v><v>262.424482
</Item></MIDataObject>
debug->提取计算结果
log->虚拟地理环境分布式模型测试-高斯大气污染扩散模型 结束
```

```
</v><v>-28.72</v></r><r index="739"><v>1501915</v><v>3566972</v><v>-29.89</v></r><r index="740"><v>1506901</v><v>3556525</v><v>-37.7
</Item></MIDataObject>
LOG->提取数据结束！
LOG->[illegible]用户更新客户端数据！
[illegible]WATER 模型示例运行结束
[illegible]大学 GISC YGE GROUP
LOG->Compute命令结束执行成功返回。
```

Figure 19. Model execution in the cloud environment.

model description, model encapsulation and cloud deployment. The major achievements are summarised as follows:

(1) A systematic analysis was conducted on the characteristics of geographical analysis models, especially the difficulties in resource sharing due to heterogeneity. An open environment for the sharing of geographical analysis models based on cloud computing architecture was designed and developed, which can effectively integrate distributed model resources and computing resources, bridging the gap between model providers, computing resource providers and model users.
(2) Several key strategies were designed for model description, model encapsulation, model deployment and transparent access, to lay a foundation for environment construction. The results indicate that these well-designed strategies can meet the requirements for model sharing, the unified description of models' data and execution behaviour, as well as operating environment-oriented encapsulation and automated cloud deployment and operation.
(3) A relevant experimental environment was established and experiments were conducted to verify the functionality of the framework. The results of the experiments show that this open environment was designed with strong adaptability and extendibility to assist with the sharing of geographical models.

However, as the environment is only a prototype, future research are needed in several aspects:

(1) More efficient tools are needed for implementing the three strategies. For example, the current encapsulation tool can only encapsulate models with text file for the input and output, and requires manual interaction of its users.
(2) Many different standards (e.g. the OGC Geography Markup Language, WPS) have been established for geographical information sharing. The compatibility of these standards lays a critical foundation for the successful promotion of the strategies proposed in this article.

(3) 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.
(4) Computing resources on the Internet have various qualities of service which has not been seriously considered in this research. Thus, spatiotemporal principles (Yang *et al.* 2011) are required to optimise the computing resources; most of all, the execution time and space correlation properties of geographical models, calculating resources allocation by intelligently scheduling model resources, computing resources and data resources have to be investigated to provide optimal services.

Acknowledgements

We appreciate the detailed suggestions and comments from the editor and the anonymous reviewers. The work described in this article was supported by the Key Program of National Natural Science Foundation of China (Grant No. 40730527), the National Natural Science Foundation of China (Grant No. 41001223, Grant No.41101439) and the open fund from the Guangdong Key Laboratory for Urbanization and Geo-simulation in Sun Yat-sen University.

References

Abdelmoty, A.I., *et al.*, 2005. A critical evaluation of ontology languages for geographic information retrieval on the Internet. *Journal of Visual Languages & Computing*, 16 (4), 331–358.

Adams, B., 2009. Conceptual Space Markup Language (CSML): towards the cognitive semantic web. In Processing of IEEE International Conference on Semantic Computing, 253–260.

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

Ahuja, L.R., Ascough, J.C., and David, O., 2005. Developing natural resource models using the object modeling system: feasibility and challenges. *Advances in Geosciences*, 4, 29–35.

Argent, R.M., 2004. An overview of model integration for environmental applications-components, frameworks and semantics. *Environmental Modelling & Software*, 19 (3), 219–234.

Argent, R.M., *et al.*, 2006. Comparing modelling frameworks: a workshop approach. *Environmental Modeling & Software*, 21 (7), 895–910.

Armbrust, M., 2010. Above the clouds: a Berkeley view of cloud computing. *Communications of the ACM*, 53 (4), 50–58.

Ascough, J.C., *et al.*, 2002. Multicriteria spatial decision support systems: overview, applications, and future research directions. *Systems Research*, 1, 175–180.

Athanasis, N., *et al.*, 2009. Towards a semantics-based approach in the development of geographic portals. *Computers & Geosciences*, 35 (2), 301–308.

Bennett, D.A., 1997. A framework for the integration of geographical information systems and model base management. *International Journal of Geographical Information Science*, 11 (4), 337–357.

Bhatt, M., Rahayu, W., and Sterling, G. 2004. *sedOnto: a web enabled ontology for synthetic environment representation based on the SEDRIS Specification* [online]. Available from: http://cindy.informatik.uni-bremen.de/cosy/staff/bhatt/seer/sedOnto-SIW-04.pdf [Accessed 5 May 2012].

Bhatt, M., Rahayu, W., and Sterling, G., 2005. Synthetic environment representational semantics using the web ontology language. *Lecture Notes in Computer Science*, 3578, 245–250.

Blind, M., and Gregersen, J.B., 2005. Towards an open modelling interface (OpenMI) the Harmon IT project. *Advances in Geosciences*, 4, 69–74.

Britz, W., Dominguez, I.P., and Heckelei, T., 2010. A comparison of CAPRI and SEAMLESS-IF as Integrated Modelling System. *Environmental and agricultural modelling*, 3, 257–274.

Buyya, R., Chee, S.Y., and Venugopal, S., 2008. Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities [online]. Available from: http://www.cloudbus.org/ ~ raj/papers/hpcc2008_keynote_cloudcomputing.pdf [Accessed 10 November 2011].

Campbell, A.P., and Hummel, J.R., 1998. The dynamic information architecture system: an advanced simulation framework for military and civilian applications [online]. *Processing of the 1998 advanced simulation technologies conference*, 25–27 September 2008. Available from: http://www.osti.gov/energycitations/servlets/purl/8876-7lKVjR/webviewable/8876.pdf [Accessed 10 November 2011].

Campos, J., and Hull, G., 2004. TCRS – a methodology and tool set for specifying data content [online]. Available from: http://www.sedris.org/presentation/04S-SIW-127.pdf [Accessed 5 May 2012].

Chen, M., *et al.*, 2009a. Semantic guided geographic conceptual modeling environment based on icons. *Geographical Research*, 28 (3), 705–715.

Chen, M., *et al.*, 2009b. Geographic problem-solving oriented data representation model. *Geo-Information Science*, 11 (3), 333–337.

Chen, M., *et al.*, 2011. A visualization method for geographic conceptual modelling. *Annals of GIS*, 17 (1), 15–29.

Chen, S.P., 1998. *The earth system science*. Beijing: China Science and Technology Press.

Craglia, M., *et al.*, 2008. Next-generation digital earth: a position paper from the Vespucci initiative for the advancement of geographic information science. *International Journal of Spatial Data Infrastructures Research*, 3, 146–167.

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

CSDMS Working Group. 2004. Community surface dynamics modeling system (CSDMS) Science Plan[online]. Available from: http://csdms.colorado.edu/mediawiki/images/CSDMS_Science_Plan_Aug04.pdf [Accessed 3 May 2012].

Diaz, L., *et al.*, 2008. An open service network for geospatial data process. *In*: *2008 Free and open source software for Geospatial (FOSS4G) conference*, 29 September–4 October 2008, Cape Town, South Africa, 410–420.

Donato, P.D., 2010. Geospatial semantics: a critical review. *Lecture Notes in Computer Science*, 6016, 528–544.

Feng, M., Liu, S., Jr, N.H.E., and Yin, F., 2009. Distributed geospatial model sharing based on open interoperability standards. *Journal of Remote Sensing*, 13 (6), 1060–1066.

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

Fook, K.D., *et al.*, 2009. Geoweb services for sharing modelling results in biodiversity networks. *Transactions in GIS*, 13 (4), 379–399.

Galton, A., 2003. Desiderata for a spatio-temporal geo-ontology. *Lecture Notes in Computer Science*, 2825, 1–12.

Goodchild, M.F., Parks, B.O., and Steyaert, L.T., 1993. *Environmental modeling with GIS*. New York, NY: Oxford University Press.

Goodchild, M.F, 2005. GIS and modeling overview. *In*: D.J. Maguire, M. Batty, and M. F. Goodchild, eds. *GIS, spatial analysis, and modeling*. Redlands: ESRI Press, 1–17.

Goodchild, M.F, 2009. Geographic information systems and science: today and tomorrow. *Annals of GIS*, 15 (1), 3–9.

Goodchild, M.F., 2010. Twenty years of progress: GIScience in 2010. *Journal of Spatial Information Science*, 1, 3–20.

Geller, G., 2010. The ecological model web concept: a consultative infrastructure for researchers and decision makers using a Service Oriented Architecture [online]. Available from: http://meetingorganizer.copernicus.org/EGU2010/EGU2010-7705.pdf [Accessed 4 May 2012].

Geller, G.N. and Melton, F., 2008. Looking forward: applying an ecological model web to assess impacts of climate change. *Biodiversity*, 9, 3–4.

Geller, G.N., and Turner, W., 2007. The model web: a concept for ecological forecasting. *In*: *IEEE international geoscience and remote sensing symposium*, 23–28 July 2007, Barcelona, Spain.

Granell, C., Diaz, L., and Gould, M., 2010. Service-oriented applications for environmental models: reusable geospatial services. *Environmental Modelling & Software*, 25 (2), 182–198.

Grimshaw, A., *et al*., 2009. An open grid services architecture primer. *Computer*, 42 (2), 27–34.

Hill, C., *et al*., 2004. The architecture of the earth system modeling framework. *Computing in Science & Engineering*, 6 (1), 18–28.

Hillyer, C., *et al*., 2003. The ModCom modular simulation system. *European Journal of Agronomy*, 18 (3), 333–343.

Hunt, G., and Brubacher, D., 1999. Detours: binary interception of Win32 functions. *In Proceedings of the 3rd USENIX Windows NT symposium*, 12–15 July 1999, Seattle, WA, 135–143.

Kuhn, W., 1994. Defining semantics for spatial data transfers. *In*: *Processing of sixth international symposium on spatial data handling*, 5–9 September 1994, Scotland, Edinburgh, 937–987.

Kuhn, W., 2001. Ontologies in support of activities in geographical space. *International Journal of Geographical Information Science*, 15 (7), 613–631.

Kuhn, W., 2002. Modeling the semantics of geographic categories through conceptual integration. *Geographic Information Science*, 2478, 108–118.

Kuhn, W., 2003. Semantic reference systems. *International Journal of Geographical Information Science*, 17 (5), 405–409.

Kuhn, W., 2005. Geospatial semantics: why, of what, and how? *Journal on Data Semantics*, 3, 587.

Larson, J.W., Norris, B., and Ong, E.T., 2004. Components, the common component architecture, and the climate/weather/ocean community. *In*: *84th American meteorological society annual meeting*. American Meteorological Society, Seattle, Washington.

Leavesley, G.H., *et al*., 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, Soil Pollution*, 90 (1–2), 303–311.

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), 173–187.

Lin, H., Huang, F.R., and Lü, G.N., 2009. Development of virtual geographic environments and the new initiative in experimental geography. *Acta Geographica Sinica*, 64 (1), 7–20.

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

Macchi, W., and Sims, E., 2002. Creating interchangeable human models Using the SEDRIS DRM and H-Anim. *Fall 2002 simulation interoperability workshop*, 20 August 2002, Orlando, Florida.

Maguire, D.J., Batty, M., and Goodchild, M.F., 2005. *GIS, spatial analysis, and modeling*. Redlands, CA: ESRI Press.

Mark, D.M., Smith, B., and Tversky, B., 1999. Ontology and geographic objects: an empirical study of cognitive categorization. *Lecture Notes in Computer Science*, 1661, 283–298.

Matthies, M., Giupponi, C., and Ostendorf, B., 2007. Environmental decision support systems: current issues, methods and tools. *Environmental Modelling and Software*, 22 (2), 123e127.

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

Maxwell, T. and Costanza, R., 1997b. A language for modular spatio-temporal simulation. *Ecological Modelling*, 103 (2–3), 105–133.

Mennis, J.L., Peuquet, D.J., and Qian, L.J., 2000. A conceptual framework for incorporating cognitive principles into geographical database representation. *International Journal of Geographical Information Science*, 14 (6), 501–520.

Muhanna, W.H., 1993. An object-oriented framework for model management and DSS development. *Decision Support Systems*, 9 (2), 217–229.

Morozov, I., Reilkoff, B., and Chubak, G., 2006. A generalized web service model for geophysical data processing and modeling. *Computer and GeoSciences*, 32 (9), 1403–1410.

Moore, R.V., and Tindall, C.I., 2005. An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy*, 8 (3), 279–286.

Nativi, S., Mazzetti, P., and Geller, G.N., 2012. Environmental model access and interoperability: The GEO Model Web initiative [online]. Available from: http://www.sciencedirect.com/science/article/pii/S1364815212000898 [Accessed 3 May 2012].

NRC, 2010. *The rise of games and high performance computing for modeling and simulation*. Washington, DC: The National Academies Press.

Open GIS Consortium (OGC), 2004. OpenGIS web services architecture [online]. Available from: http://portal.opengeospatial.org/files/?artifact_id = 1320 [Accessed 10 November 2011].

Open GIS Consortium (OGC), 2008. Web processing service (WPS) Specification [online]. Available from: http://portal.opengeospatial.org/files/?artifact_id = 13149 [Accessed 10 November 2011].

Oxley, T., *et al.*, 2004. Integrated modelling and decision-support tools: a Mediterranean example. *Environmental Modelling and Software*, 19 (11), 999–1010.

Rafanelli, M., 2003. *Multidimensional databases: problems and solutions.* Hershey, PA: Idea Group Publishing.

Raskin, R.G. and Pan, M.J., 2005. Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). *Computers & Geosciences*, 31 (9), 1119–1125.

Reed, M., Cuddy, S.M., and Rizzoli, A.E., 1999. A framework for modeling multiple resource management issues-an open modelling approach. *Environmental Modelling & Software*, 14 (6), 503–509.

Rizzoli, A.E., and Young, W.J., 1997. Delivering environmental decision support systems: software tools and techniques. *Environmental Modelling and Software*, 12 (2), 237–249.

Smart, P., Abdelmoty, A.I., and Jones, C.B., 2004. An evaluation of geo-ontology representation languages for supporting web retrieval of geographical information [online]. Available from: http://www.geo-spirit.org/publications/psmart-gisruk04-final.pdf [Accessed 5 May 2012].

Siddhisena, B., Warusawithana, L., and Mendis, M., 2011. Next generation multi-tenant virtualization cloud computing platform [online]. *Processing of 13th international conference on advanced communication technology (ICACT)*, 13–16 February 2011, 405–410. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5745841 [Accessed 11 November 2011]

Simunich, K.L., *et al.*, 2002, Dynamic information architecture system (DIAS): multiple model simulation management [online]. *Processing of the 2nd federal interagency hydrologic modeling conference.* Available from: http://www.ipd.anl.gov/anlpubs/2002/05/43066.pdf [Accessed 10 November 2011].

The Higher ED CIO, 2011. *Cloud or Not: 5 characteristics of cloud computing* [online]. Available from: http://blog.thehigheredcio.com/2011/02/19/characteristics-cloud-computing/ [Accessed 19 February 2012].

Tsou, M., and Buttenfield, B.P., 2003. A dynamic architecture for distributing geographic information services. *Transactions in GIS*, 6 (4), 355–381.

Ungerer, M.J., and Goodchild, M.F., 2002. Integrating spatial data analysis and GIS: a new implementation using the Component Object Model (COM). *International Journal of Geographical Information Science*, 16 (1), 41–53.

Van Ittersum, M.K., *et al.*, 2008. Integrated assessment of agricultural systems-a component-based framework for the European union (SEAMLESS). *Agricultural Systems*, 96 (1–3), 150–165.

Valcke, S., Guilyardi, E., and Larsson, C., 2006. PRISM and ENES: a European approach to Earth system modelling. *Concurrency and Computation: Practice and Experience*, 18 (2), 247–262.

Visser, U., *et al.*, 2002. Ontologies for geographic information processing. *Computers & Geosciences*, 28 (1), 103–117.

Watson, F.G.R., and Rahman, J.M., 2004. Tarsier: a practical software framework for model development, testing and deployment. *Environmental Modelling & Software*, 19 (3), 245–260.

Yang, C., *et al*., 2008. Distributed geospatial information processing: sharing earth science information to support digital earth. *International Journal of Digital Earth (IJDE)*, 1 (3), 259–278.

Yang, C., *et al*., 2010. Geospatial cyberinfrastructure: past, present and future. *Computers, Environment, and Urban Systems*, 34 (4), 264–277.

Yang, C., *et al*., 2011. Using spatial principles to optimize distributed computing for enabling the physical science discoveries. *Proceedings of the National Academy of Sciences*, 108 (14), 5498–5503.

Yue, T.X., 2003. *Resources and environment models manual*. Beijing: Science press.

Zuo, D.K., 2009. *Dictionary of modern geography.* Beijing: Commercial Press.