ISPRS Journal of Photogrammetry and Remote Sensing 114 (2016) 258–273

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

# Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment

Songshan Yue [a,b,c], Min Chen [a,b,c,d], Yongning Wen [a,b,c,*], Guonian Lu [a,b,c]

[a] Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China
[b] State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing 210023, China
[c] Key Laboratory of Virtual Geographic Environment (Nanjing Normal University), Ministry of Education, Nanjing 210023, China
[d] Institute of Space and Earth Information Science, The Chinese University of Hong Kong, Shatin, Hong Kong, China

ARTICLE INFO

*Article history:*
Received 26 November 2014
Received in revised form 29 October 2015
Accepted 5 November 2015
Available online 23 November 2015

*Keywords:*
Geo-analysis models
Sharing
Integration
Encapsulation
Open web environment

ABSTRACT

Earth environment is extremely complicated and constantly changing; thus, it is widely accepted that the use of a single geo-analysis model cannot accurately represent all details when solving complex geo-problems. Over several years of research, numerous geo-analysis models have been developed. However, a collaborative barrier between model providers and model users still exists. The development of cloud computing has provided a new and promising approach for sharing and integrating geo-analysis models across an open web environment. To share and integrate these heterogeneous models, encapsulation studies should be conducted that are aimed at shielding original execution differences to create services which can be reused in the web environment. Although some model service standards (such as Web Processing Service (WPS) and Geo Processing Workflow (GPW)) have been designed and developed to help researchers construct model services, various problems regarding model encapsulation remain. (1) The descriptions of geo-analysis models are complicated and typically require rich-text descriptions and case-study illustrations, which are difficult to fully represent within a single web request (such as the *GetCapabilities* and *DescribeProcess* operations in the WPS standard). (2) Although Web Service technologies can be used to publish model services, model users who want to use a geo-analysis model and copy the model service into another computer still encounter problems (e.g., they cannot access the model deployment dependencies information). This study presents a strategy for encapsulating geo-analysis models to reduce problems encountered when sharing models between model providers and model users and supports the tasks with different web service standards (e.g., the WPS standard). A description method for heterogeneous geo-analysis models is studied. Based on the model description information, the methods for encapsulating the model-execution program to model services and for describing model-service deployment information are also included in the proposed strategy. Hence, the model-description interface, model-execution interface and model-deployment interface are studied to help model providers and model users more easily share, reuse and integrate geo-analysis models in an open web environment. Finally, a prototype system is established, and the WPS standard is employed as an example to verify the capability and practicability of the model-encapsulation strategy. The results show that it is more convenient for modellers to share and integrate heterogeneous geo-analysis models in cloud computing platforms.


## 1. Introduction

A substantial number of extremely complicated and constantly evolving geo-phenomena and geo-processes occur in Earth's environment (Serreze, 2011; Albanesi and Albanesi, 2014). Geo-analysis models are considered important tools for simulating

* Corresponding author at: Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China. Tel.: +86 13913859225; fax: +86 02585898270.
*E-mail addresses:* wenyongning@njnu.edu.cn, wenyn@msn.com (Y. Wen).

http://dx.doi.org/10.1016/j.isprsjprs.2015.11.002
0924-2716/

these geo-phenomena and geo-processes (Parsons, 2011; Lin et al., 2013a, 2013b; Sudo et al., 2013). Through years of research, numerous geo-analysis models (on various geo-problems, study areas, spatiotemporal scales) have been built in a variety of disciplines (e.g., Goodchild et al., 1996; Basnyat et al., 2000; Dickinson et al., 2006; Todorova et al., 2010; Dennis et al., 2012; Li et al., 2013, 2015; Zhang et al., 2014, 2015a; Zhu et al., 2015; Yin et al., 2015). However, when simulating synthetic geo-processes and complicated geo-phenomena, the knowledge of a single discipline can hardly address all situations in the environment. The study of sharing and integrating geo-analysis models has been shown as an efficient method to connect researchers across disciplines and to further solve comprehensive geo-problems, especially in interdisciplinary research (Argent, 2004; Argent et al., 2006; Chen et al., 2011; Lin et al., 2015). Sharing and integrating geo-analysis models in an open web environment can also reduce the cost of applying geo-analysis models in practice, thus promoting modelling research (Sancho-Jiménez et al., 2008; Nativi et al., 2013).

To share and integrate heterogeneous geo-analysis models, several problems regarding the heterogeneities of various models must be considered. Because geo-analysis models are developed using different technical approaches by individual researchers or research groups, they are often built on different platforms (e.g., Windows and Linux), coded in different programming languages (e.g., FORTRAN, C/C++, Python) and designed with different user interfaces (e.g., console command line, execution file, dynamic link library). Moreover, it is generally difficult for researchers to understand models from other disciplines. To reduce the heterogeneities of geo-analysis models, encapsulation (also referred to as wrappers in the software development field) studies are necessary.

To date, several model sharing and integration methods have been studied using a variety of theoretical and technological approaches. Some methods are designed to integrate specific models: the coupling of HEC-RAR with MODFLOW to simulate stream-aquifer interactions (Rodriguez et al., 2008); SWAP and MODEFLOW-2000 for modelling groundwater dynamics (Xu et al., 2012); and land-use forecasting models and hydrologic models for improving land-use decision support (McColl and Aggett, 2007). Moreover, a range of frameworks and platforms for sharing and integrating geo-analysis models has been proposed and developed to support dynamic modelling. Examples include the Spatial Modeling Environment (SME) (Maxwell and Costanza, 1997a, 1997b), ModCom (Hillyer et al., 2003), the Open Modeling Interface (OpenMI) (Blind and Gregersen, 2005 and Moore and Tindall, 2005), the European Union's Program for Integrated Earth System Modelling (PRISM) (Valcke et al., 2006), the Common Agricultural Policy Regional Impact modelling system (CAPRI) (Britz et al., 2010), and the Community Surface Dynamics Modeling System (CSDMS) (Peckham and Hutton, 2009). Different modelling frameworks employ various model encapsulation methods. Some frameworks use re-programming (Succar, 2009), while others use input/output data conversion (Pullar and Springer, 2000), and some are based on command calling (Lei et al., 2011).

However, most of these integration frameworks are field specific, and few of these frameworks support the sharing and integration of models over the Internet. Trends in the Internet and in software engineering development over the past 15–20 years have promoted a range of web-based applications that allow users to more easily communicate with each other and access resources. Because geo-analysis models are typically distributed globally and are owned by researchers in multiple disciplines, sharing and integrating geo-analysis models in an open web environment can facilitate the access and use of geo-analysis models (Laniak et al., 2013; Chen et al., 2014). Through this open web environment, model providers can offer their own models, and model users can reuse and integrate these models. Service-oriented Architecture (SOA), which is widely used in the information-sharing and business-processing fields, has been used to integrate model data and geo-analysis models worldwide (Erl, 2008).

Based on the web architecture, the Open GIS Consortium (OGC) has drafted several specifications concerning data and modelling services, such as the Web Mapping Service (WMS), Web Feature Service (WFS), and Web Processing Service (WPS). Based on the WPS standard, much software has been developed to help modellers build model services, such as the PyWPS (Python Web Processing Service), 52°North project and QGIS (Quantum GIS). In addition, Simple Object Access Protocol and Web Service Description Language (SOAP/WSDAL) and 'Model Web' have been studied in terms of the sharing and integration of geo-analysis models (Geller and Turner, 2007). Cloud computing, which is commonly viewed as an efficient method of sharing resources in an open web environment, has also been investigated for geo-analysis modelling services (Wen et al., 2013). Based on these web-based model-services, scientific workflows and business process workflows can be constructed (such as Kepler, WS-BPEL). However, the lack of user control and interactivity during execution appears to limit the application in using model-services (Granell et al., 2013; Zhang et al., 2015b).

Cloud computing can provide massive storage and computational resources for the execution of geo-analysis models. When using a cloud-computing platform, the original model resources are moved from a personal computer to transparent computation servers. Although cloud computing and other web processing architectures provide an open web environment to share resources (model services can also be treated as resources), several existing legacy models cannot be directly uploaded to a server node and used for a model service. The encapsulation work of geo-analysis models is important, and the following difficulties regarding this work are observed.

(1) Heterogeneity of geo-analysis model descriptions. In addition to the execution heterogeneities, substantial variations occur in multi-disciplinary geo-analysis models caused by the different research contexts of disciplines (modellers have difficulties in understanding and employing geo-analysis models of another discipline). A structured model-description method is required to meet the demands of model discovery by model users and model encapsulation by model providers.
(2) Differences between model execution behaviour and web services. Because geo-analysis models typically involve several computation states that partially reflect real geo-processes, model users must act according to the specific requirements in these states. However, the basic architecture of a web service is designed as a simple request–response process, which is in contrast to the execution behaviour of geo-analysis models. Therefore, the description of model execution behaviour that meets the demands of web service architecture should also be considered in model encapsulation.
(3) The deployment information of model-services must be communicated between model providers and model users. In an open web environment, geo-analysis models will be offered and deployed by model providers, and model users employ these models within the same web environment. However, model users who want to use a geo-analysis model and want to copy the model service into another computer still encounter problems (e.g., they cannot access the hardware/software dependencies information). Therefore, model encapsulation should contain sufficient information (flexible and extendable) for modellers from different countries with different research backgrounds.

This study does not focus on building a model-integration framework or model service architecture to solve all the problems in model sharing and integration; instead, we attempt to develop a strategy to encapsulate geo-analysis models that can further support the sharing and integration of models in an open web environment. As shown in Fig. 1, the encapsulation results can be published as model services after encapsulating the original codes or execution programmes. Web Service technologies (for example, the WPS standard) can be used to publish model services, and cloud computing platforms can contain these services. Furthermore, the main idea of this study is that model providers should be able to encapsulate their original geo-analysis models by implementing a set of model encapsulation interfaces. Model users can obtain sufficient information regarding encapsulated models and related resources through these interfaces. The proposed service-oriented encapsulation strategy will be used to study the description method for heterogeneous geo-analysis models. The method for encapsulating the model-execution program to model services and for describing model-service deployment information is included based on the model descriptions. Based on this encapsulation strategy for geo-analysis models in an open web environment, model providers can organise model descriptions, model-execution behaviour and model-deployment information in a structured and extendable manner. Moreover, model users can access models, prepare model input data, and use model services more easily; thus, the difficulty of sharing and integrating heterogeneous geo-analysis models in an open web environment can be reduced.

The remainder of this article is structured as follows. The basic idea behind the model encapsulation strategy is introduced in Section 2. In Section 3, the detailed model-encapsulation method with the design of the model-description interface, model-execution interface, and model-deployment interface is explained. Section 4 introduces the realisation of a prototype system and related experiments. Finally, the conclusions and a discussion of the study are presented in Section 5.

## 2. Basic idea of the model-encapsulation strategy

The proposed model encapsulation strategy focussed on the sharing process between model providers/users and model services. As shown in Fig. 2, model providers ought to describe the models that they want to publish; therefore, model users can more easily find and use models by searching with descriptive keywords. Regarding this process, the WPS standard includes items to support "a human readable description" (according to the Web Processing Service 2.0 Standards Working Group, http://www.opengeospatial.org/projects/groups/wps2.0swg). However, the WPS standard only permits a relatively static description (i.e., once a model service is launched, the description information is assigned and can be difficult to modify). Geo-analysis models are developed by individual researchers and research groups from around the world. The descriptions of geo-analysis models are complicated (e.g., presented by using different languages) and typically require rich-text descriptions (e.g., videos and animation pictures) and case-study illustrations, which are difficult to represent when using a single web query service (e.g., *GetCapabilities* and *DescribeProcess* requests in the WPS standard).

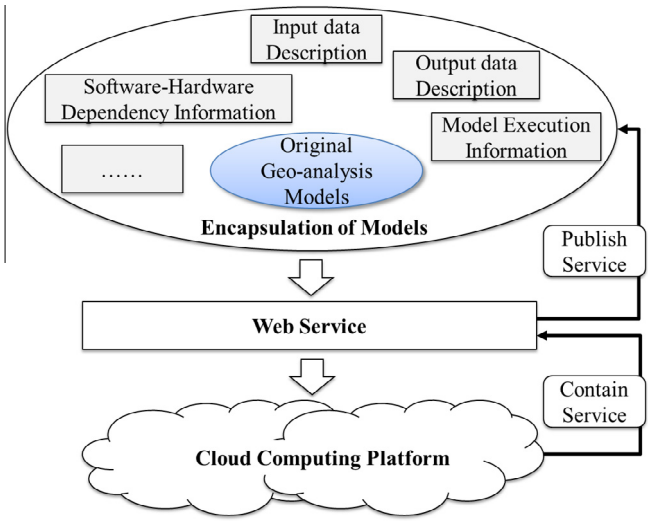



**Fig. 1.** Encapsulation of geo-analysis models and model service.

In Fig. 2, model-related resources (data service, model service, and computer service) are treated as disparate web cloud nodes, and the specific storage location is typically transparent for users. Additionally, numerous web transmission methods are available to users for accessing these resources (e.g., HTTP, P2P, WebSocket). The proposed encapsulation strategy is not tightly bound to any web architecture or web transmission method; instead, it offers a simple method for users to apply model services in practice with the interface designation. The designed interfaces can provide various approaches for model providers to fully describe models, wrap model-executing programs into model services, and deploy models into specific web nodes. Additionally, through the designed interfaces, model users can obtain model descriptions, run models with full control, and integrate different models to solve geo-problems.

Geo-analysis models should be encapsulated and published according to the specific technology architecture (such as the ASP.Net platform or the JAVA platform) so that model users can correctly prepare input data and utilise models through corresponding web requests and response commands. Considering the working process of model providers and model users, the proposed encapsulation strategy consists of three basic interfaces: the model-description interface, the model-execution interface and the model-deployment interface.

(1) The model-description interface is implemented to unambiguously and fully describe model information. Using the model-description interface, model providers can describe the models in a flexible and structured manner (with extendable language versions and foreign well-known website linkages), and model users can obtain this descriptive information. With the model-description interface, the heterogeneity of geo-analysis model descriptions can be reduced.

(2) The model-execution interface is designed to explain the models' input and output (IO) data and execution behaviour. The model-execution interface includes the model data description method and the model execution behaviour description method and is typically combined with programming tasks. The model data description method is used to reduce the heterogeneity of the model data, and the model execution behaviour description method is used to convert complex model computation states into web services. By employing the model data description method and the model execution behaviour description method, heterogeneous models can be more easily encapsulated as model services. In addition, model users can obtain information on model execution states and take action according to the state using the model-execution interface.

(3) The model-deployment interface is used to describe model-service deployment information, such as the web resource location of the computer that contains the model service

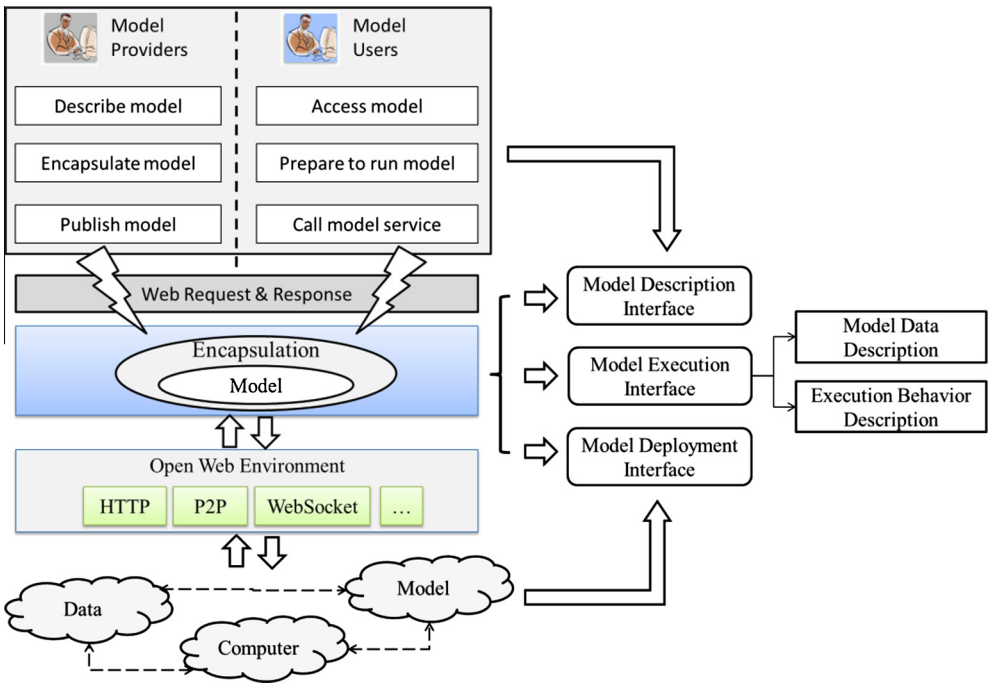



**Fig. 2.** Concept of service-oriented geo-analysis model-encapsulation strategy.

and the model execution dependencies. To utilise geo-analysis models as model services, model providers should offer sufficient "install" information to deploy models on computers. In addition, model users need to know the deployed location so that model services can be invoked. Using the model deployment interface, model providers can more conveniently publish geo-analysis models as services, and model users can use these model services with additional execution background information, which can facilitate model users' understanding of model services.

Through the three main encapsulation interfaces, the collaboration between model providers and model users becomes easier, thus reducing the difficulty of sharing and integrating geo-analysis models in an open web environment. Model encapsulation focuses on the working process in which model providers offer information that is sufficient for model users to understand and use the shared model resources. Model encapsulation is independent of specific technology approaches and can be used when implementing model services (for example, the WPS standard can be used to more easily implement the proposed model encapsulation interfaces by extending some request methods of the WPS standard).

## 3. Encapsulation strategy for heterogeneous geo-analysis models

### 3.1. Description interface of geo-analysis models

When solving synthetic environment problems by integrating multiple geo-analysis models, the first step is to find appropriate models according to the modelling objective. Geo-analysis models have been developed by a variety researchers or research groups; they are often dispersed across the web environment. Therefore, model discovery is a complex issue because the volume and diversity of geo-analysis models grow exponentially. The ambiguity of search keywords leads to difficulty in finding models of interest because typical concepts in one discipline may be obscure to researchers in other disciplines. The flexibility and variety of model descriptions hinders model users from understanding models because researchers and research groups are from different countries and speak different languages. Therefore, complete and unambiguous descriptions of geo-analysis models are required.

Category services can facilitate locating resources of interest (Bergner, 2007). In addition, some model-integration frameworks have already been used to construct various types of model category systems, which are typically developed using a tree structure (Peckham and Hutton, 2009). However, building a comprehensive category system to describe all categories of models is difficult because environmental modelling research is changing, and more geo-analysis models are emerging. Regarding these problems, this study does not focus on building a new model category system; rather, we attempt to design a management solution to integrate different categories of systems.

In the software engineering field, language packages and plug-in architectures have been proposed and have been shown to be efficient methods of integrating software representation interfaces and application modules. Borrowing from these basic concepts, the proposed model-description interface (which is represented as the *IAttributeSet*, as shown in Fig. 3) consists of two main item collections: *CategoryCollection* and *LocalAttributeCollection*. Both of these collections are extendable: *CategoryCollection* is a set of the *ModelCategory* collection, and *LocalAttributeCollection* is a set of the *ModelLocalAttribute* collection. The model-description interface is designed as shown in Fig. 3.

*ModelCategory* is designed to present information on the model category to which the described model belongs. The *Principle* attribute of the *ModelCategory* object defines the unified name of a model category system, and the *Path* attribute provides the level location of the described model in a model category system. As shown in Fig. 4, the MODFLOW model is categorised in the Community Surface Dynamics Modeling System (CSDM) system (*Principle*) and belongs to the "CSDM/Terrestrial models" subcategory (*Path*).

*ModelLocalAttribute* is designed to structurally describe model declarations. *LocalizationEnum* describes the language, such as

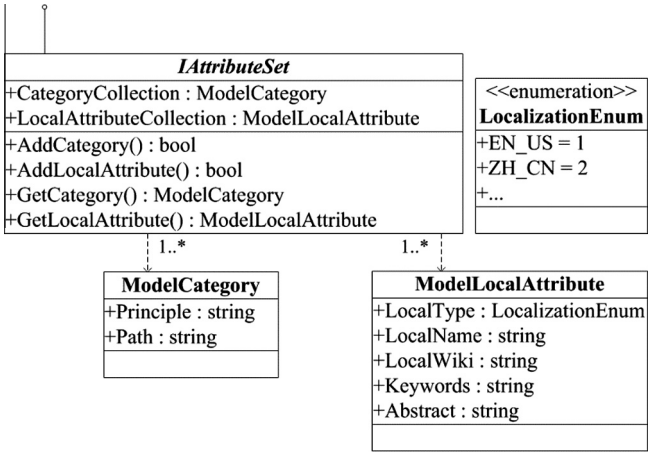



**Fig. 3.** Design of the model description interface.

English (EN_US) or Chinese (ZH_CN), that a *ModelLocalAttribute* employs (indicated by *LocalType*). Accordingly, *LocalName* (name of the model), *Keywords* (various typical descriptive words of the model), and *Abstract* (simple but useful sentences used to describe the model) in *ModelLocalAttribute* should use the language that *LocalType* declares. The *LocalWiki* attribute contains URLs (Unified Resource Links) that are related to the model, and the contents in these URLs typically explain the model more clearly, employing demonstrative usages or multi-media technology (e.g., images, video, animations). In Fig. 4, the MODFLOW model is described in two languages (English and Chinese), and the attributes of *ModelLocalAttribute* can be extracted from these two web pages. Thus, two different URLs are used to describe the MODFLOW model in English and Chinese. Furthermore, when another URL is used to explain the MODFLOW model in other languages (e.g., Japanese and Russian), a new *ModelLocalAttribute* can be implemented.

With this flexible plug-in style designation, model descriptions can be collected in a structured manner. Newly developed models, model resources in existing model-integration frameworks and web-based model service systems can be easily described with the model-description interface. Model providers and model users can use the same interface to describe models and obtain model descriptions. Based on the model-description interface, several applications can be implemented using other technological approaches, such as web crawl methods or structured query language. The basic idea of the model-description interface is easy to implement. Consequently, the difficulty of describing and discovering models is reduced.

### 3.2. Execution interface of geo-analysis models

Generally, geo-analysis models can be represented as mathematical models and can be implemented as executable programs with specific parameters and runtime platform requirements. To practically apply geo-analysis models, the input and output data must be clearly understood. Preparing input data according to the specific requirements of the different geo-analysis models is the first step in using these models (Yue et al., 2015). In addition, this preparation typically becomes substantially more complicated for model users because model data are diverse at the data format level and at the content semantic level (Li et al., 2014). To reuse and integrate heterogeneous geo-analysis models that involve heterogeneous data in a web environment, the encapsulated geo-analysis models should employ the same data description strategy so that model users can prepare model data in a consistent manner; thus, model integration could be performed with a universal data transmission method.

In addition, behind the execution program representation, geo-analysis models are the abstractions of geographical information and geo-processes. Workflows with multiple execution steps are often used to simulate real environmental running states. The ability to understand execution step information and to perform corresponding actions based on the specific execution step is typically required by model users. Model execution behaviour information should be included within the encapsulation geo-analysis models so that the original geo-process characteristics of the models can be retained; simply regarding geo-analysis models as "give input and get output" black boxes cannot meet demands in complex modelling applications.

The descriptions of the model data and model execution behaviour are similar to those obtained from Data Provenance research. Many studies regarding Data Provenance have been conducted to track the input–output relationships and geospatial analysis history in workflow systems (Alonso and Hagen, 1997; Wang et al., 2008; Malaverri et al., 2012; OGC, 2014). In addition, the Open Provenance Model (OPM) can handle spatial data provenance, including, in the context of Earth data processing systems (Tilmes and Fleig, 2008), geospatial web services and geo-processing service chains (Yue et al., 2010). In this paper, the model-execution interface is designed to provide a "descriptive level" between geo-analysis models and specific technology implementations. Based on the model-execution interface, other data provenance applications can be developed to conduct specific geo-processing workflows. Furthermore, the model-execution interface can offer sufficient information regarding model data and model execution processes, which are necessary for data provenance work. The model-execution interface contains two main description methods: the model data description method and the model execution behaviour description method.

#### 3.2.1. Model data description method

Because of the variety of research disciplines and modelling methods, different geo-analysis models have their own data requirements. The format and content of model data are diverse because of the variety of modelling disciplines and researcher designations (e.g., data in fixed formats and data that are described by flexible plain text). Considering that geo-analysis models should be implemented by a programming language, all model data are converted into variables of the employed programming languages. Using a series of basic variable types and the combination of these variable types, a flexible data description model (Universal Data eXchange model, UDX) is proposed to reduce the heterogeneity of the model data.

In Fig. 5, the Unified Modeling Language (UML) diagram of the UDX model is explained. *Node* and *Kernel* are two basic construction elements of the UDX model. *Node* controls the model data hierarchy structure, and *Kernel* controls the type of *Node*. Every *Node* has a unique *Name* and *Kernel* and possesses a *NodeChildren* that contains the number of child *Nodes* (0–*n*).

*Kernels* (*DTKernels*) can be divided into three groups based on type: *DTKValue*, *DTKValueList*, *DTKContainer*. An enumeration constant table (*KernelType*) is defined to assign a unique code to all *Kernel* types. Table 1 shows detailed information on these *KernelTypes*. *Vector2d*, *Verctor3d*, and *Vector4d* are designed to help construct these *Kernels*. Six types of values are employed in the UDX model to present basic data: *Int*, *Float*, *String*, *Vector2d*, *Vector3d*, *Vector4d*. The precision of the basic value type is not fixed. Although *DTKFloatValue* means that the value is a floating number (also called a decimal), the precision is not fixed to 32 bits or 64 bits. The precision can be assigned using other UDX *Nodes* (using a *DTKIntValue Node* to indicate the precision that the *DTKFloatValue Node* used), and the actual data can be correctly interpreted using a program that implements the UDX model.

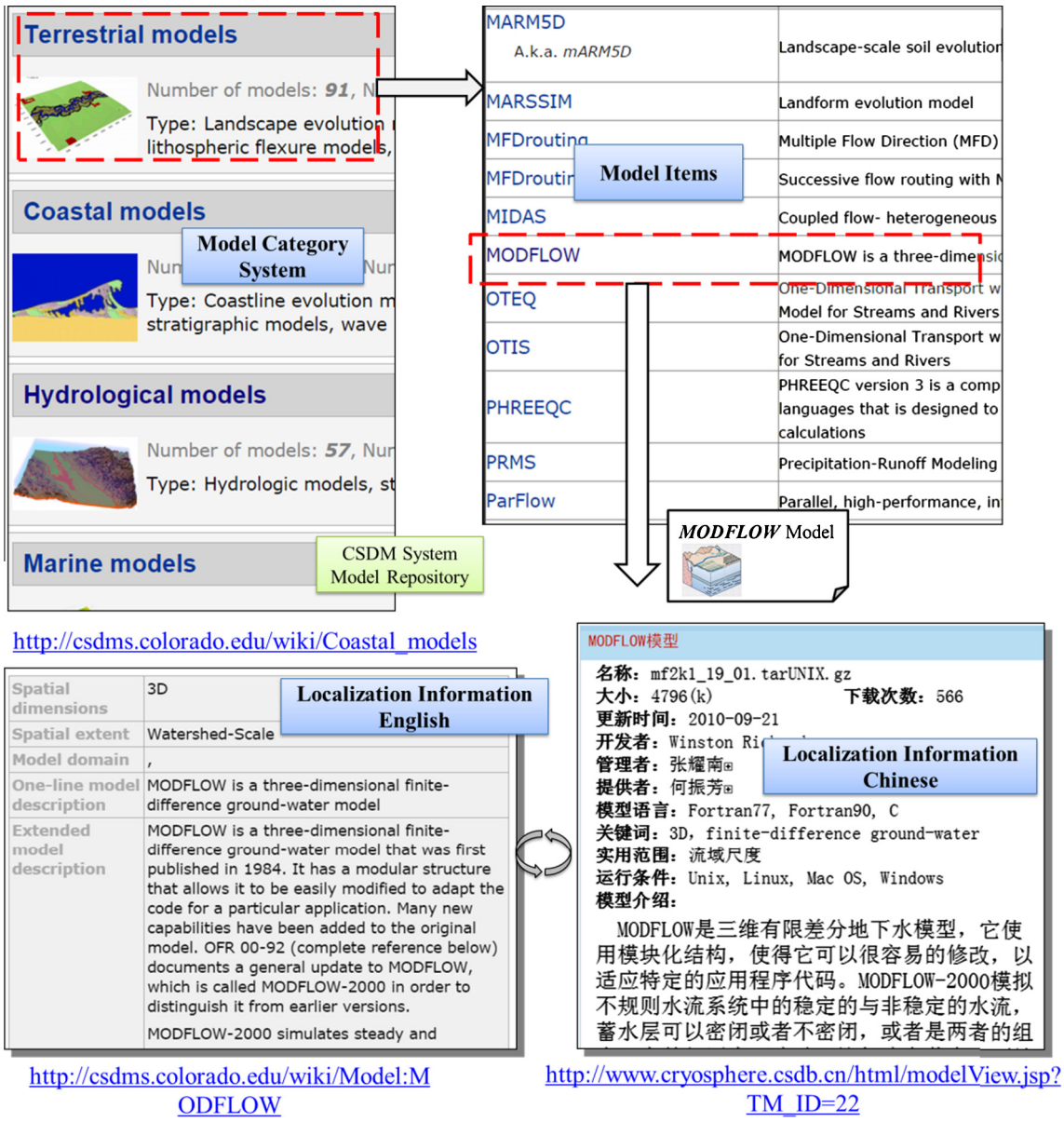



**Fig. 4.** Examples of a model category system and model localisation information.

Fig. 6 introduces the UDX data structure using Hydro Response Unit (HRU) data (Arnold et al., 1998) as an example. Watershed data should be appropriately prepared to use the Soil and Water Assessment Tool (SWAT) model, and HRU data are important input data when combined with watershed data. Fig. 6 includes two main parts: the bottom part presents the original watershed data, and the top part shows the corresponding description using the UDX model. As shown by the gears presented in the bottom of Fig. 6, the original watershed data consist of several parts organised in plain text (HRU data) and binary format (shapefile, sub basin data and sub basin's attribute data). Using the UDX model, the watershed data are described as a complicated *DTKStructure* node that is constructed using various objects, such as the Watershed_Name node (*DTKStringValue*, describing the name of the watershed), Watershed_Area node (*DTKFloatValue*, describing the total area of the watershed), Watershed_Boundary node (*DTKVector2dList*, describing the outline of the watershed), and SubbasinList node (*DTKList*, describing all sub-basins that are within the watershed). In the SubbasinList node, there is a list of Subbasin_Item nodes, and every Subbasin_Item node is developed using the same structure: Subbasin_ID (*DTKIntValue*, describing the ID of the sub-basin), Subbasin_Area (*DTKFloatValue*, describing the area of the sub-basin), Subbasin_Boundary (*DTKVector2dList*, describing the outline of the sub-basin), Subbasin_Attribute (*DTKTable*, describing all the attributes of the sub-basin). In the Subbasin_Attribute node, all child nodes should be of the type *DTKValueList*. For example, the HRU_SlopeLength node is of the type *DTKFloatList*, and it describes all HRU slope length values. Using this "construction" method, complicated model data can be fully and flexibly described.

In contrast to other intermediate data formats in data conversion fields, the UDX model is not only a data format but also a set of methods used to uniformly and completely describe model data. Because it is derived from the UDX model, there are two main implementations: UDX Data and UDX Schema. UDX Data is designed to interchange model data in their original format with

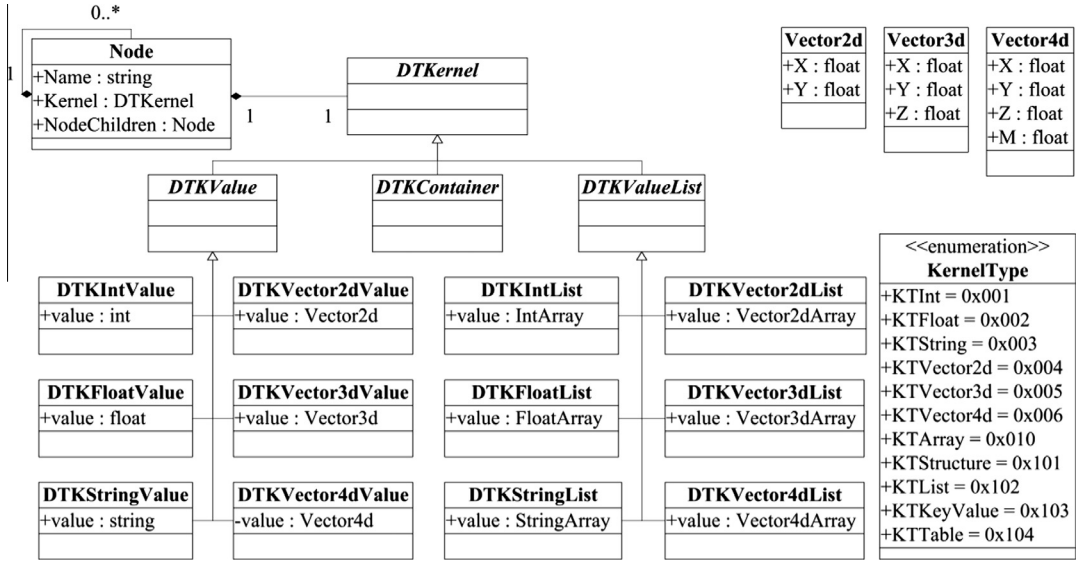



**Fig. 5.** UML diagram of the UDX model.

**Table 1**
*Kernel* type of *Node*.

| Group | Kernel type |
|---|---|
| *DTKValue* | The kernel should be limited to *DTKIntValue*, *DTKFloatValue*, *DTKStringValue*, or *Vector type (DTKVector2dValue*, *DTKVector3dValue*, *DTKVector4dValue)*. In addition, the node cannot have any child nodes; the value that the node contains must be consistent with the kernel type |
| *DTKValueList* | The *Node* has a list of values, and the values must be the same type as one of the kernel types in the *DTKValue* group (*DTKIntList*, *DTKFloatList*, *DTKStringList*, *DTKVector2dList*, *DTKVector3dList, DTKVector4dList*). Additionally, *DTKValueList* cannot have any child nodes |
| *DTKContainer* | Four types: (1) *Structure*, (2) *List*, (3) *Key-Value*, or (4) *Table*. The DTKContainer node itself does not hold any data. The specified kernel type also applies to the child nodes, and all the data should be stored in the child nodes |

the UDX model. UDX Schema is designed to provide the skeleton frame information on the model data, which should remain highly consistent with the corresponding UDX Data.

Using UDX Data, the encapsulated geo-analysis models would require data in the same organisation. In addition to preparing model input data in a variety of organisations or formats, model users can utilise model data with a common method. Moreover, the integration of different models can enable the more direct interchange of input/output data.

Independent of any specific model data, UDX Schema focuses on the full and unambiguous description of model data, which can reduce confusion and difficulty in terms of data preparation. To help model providers describe model data in a more structured manner and to help model users better understand the model (so that model data can be more easily prepared), four library attachments are constructed, as shown in Fig. 7: the unit and dimension library, the semantic (or concept) library, the spatial reference library, the data-description template library.

In Fig. 7(a), the *flow velocity* data from a *monitor section* series in a specific *area* of a river are described using a value table. For example, although "5" is a simple numerical value, it contains much more information, including that the speed units are presented in m/s and that the concept of these data is flow velocity. Therefore, the unit and dimension library provides a full measurement scale for the UDX data, the semantic (or concept) library provides accurate semantic (conceptual or ontological) information on the UDX data by borrowing the semantic systems and web from other disciplines, the spatial reference library provides coordinates or projection information for the UDX data (if the data are spatial), and the data-description template library provides an interface for interchanging with other domain-related or commonly used data formats (e.g., *Shapefile*, *Geotiff*, and *WKT*).

In Fig. 7(b), the example implementation of the UDX Schema for the *flow velocity* data of a *monitor section* series is presented. The river_edge *Node* is used to describe the spatial position of the river's edge, and the general data type of this *Node* is Shapefile, which is represented by the ShapefileData item in the data-description template library. The unit, concept and spatial reference information are contained in the UDX Schema by linking the corresponding resource items in the unit and dimension library, the semantic (or concept) library, and the spatial reference library.

The UDX model allows users to define, understand, and prepare model data for model execution, hence reducing the difficulty in data processing when encapsulating geo-analysis models.

### 3.2.2. Model execution behaviour description method

According to the above analysis, geo-analysis models often involve a substantial amount of discipline-specific knowledge and a variety of interactions with users. Amongst all the difficulties of sharing and integrating heterogeneous geo-analysis models, the complexity of model execution behaviour plays a significant role. The execution of geo-analysis models typically involves numerous intermediate "computation steps", which provide intermediate computation results or require model users to offer input data to continue the execution process (as shown in the left portion of Fig. 8, the model execution behaviour includes several mid-processes, and the execution flow can be invoked by other computation processes). Similarly, in the IT domain, workflows have been shown to be an efficient method of simplifying complicated data processing and information transmission work (Deelman et al., 2009). Hence, this study proposes a model behaviour description method: the State Machine-Event Response model (SM-ER model). In the SM-ER model, geo-analysis model execution processes are abstracted as *states*, and input/output data handling messages are abstracted as *events*. The interaction between the model execution and model users was described as *request* and *response*.

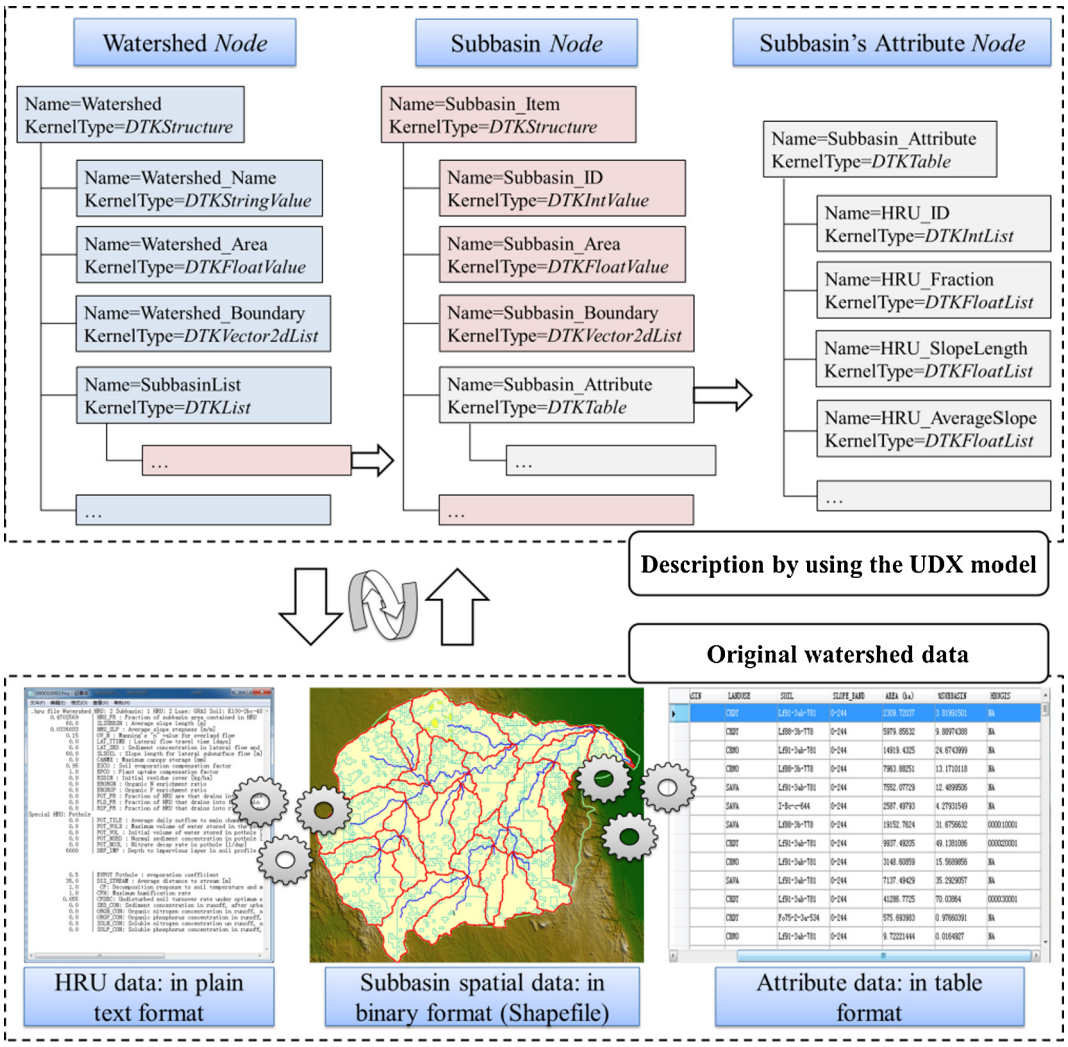


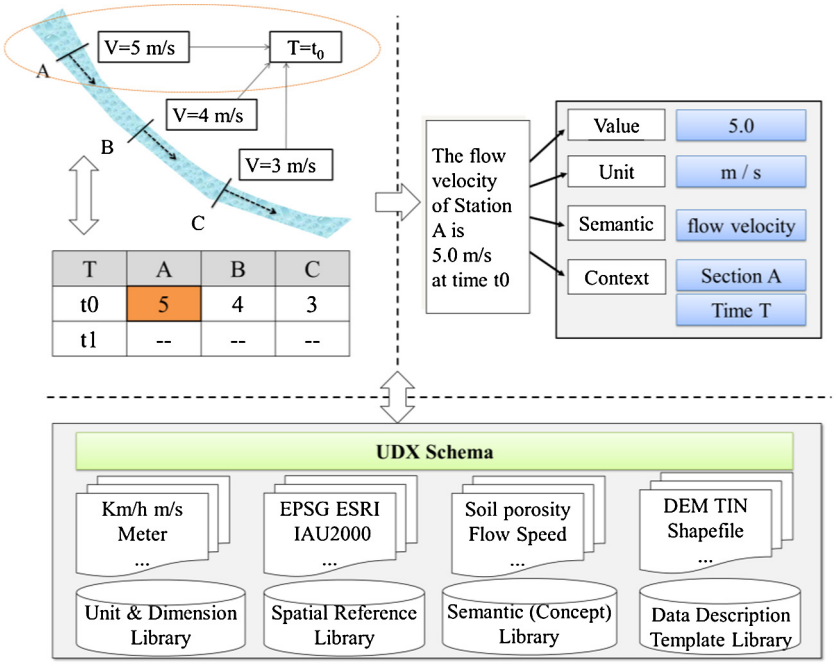

**Fig. 6.** Examples of *KernelType* in the UDX model.

The SM-ER model is illustrated in Fig. 8. The *ModelBehavior* object includes *ModelState* sets and *ModelTransition* sets. The *ModelTransition* set describes the *source state* (previous execution step), the *target state* (the next execution step) and the *condition* (which indicates whether the execution can be continued) of the *ModelTransition*. The *ModelState* set consists of a *name* and the associated *ModelEvents*, which will be used to indicate data handling messages during model execution. The *ModelEvent* includes a *name* and the numbers of *RequestData* (designed to describe external requests for data as an external input) and *ResponseData* (used to describe the data submitted to the outside as an output). If a *ModelEvent* contains *RequestData*, then the model is requesting user input (transmitted by *dataCallBack* in UDX Data). In addition, if *ResponseData* is contained in the *ModelEvent*, then the model is providing data as an output (transmitted by *data* in UDX Data).

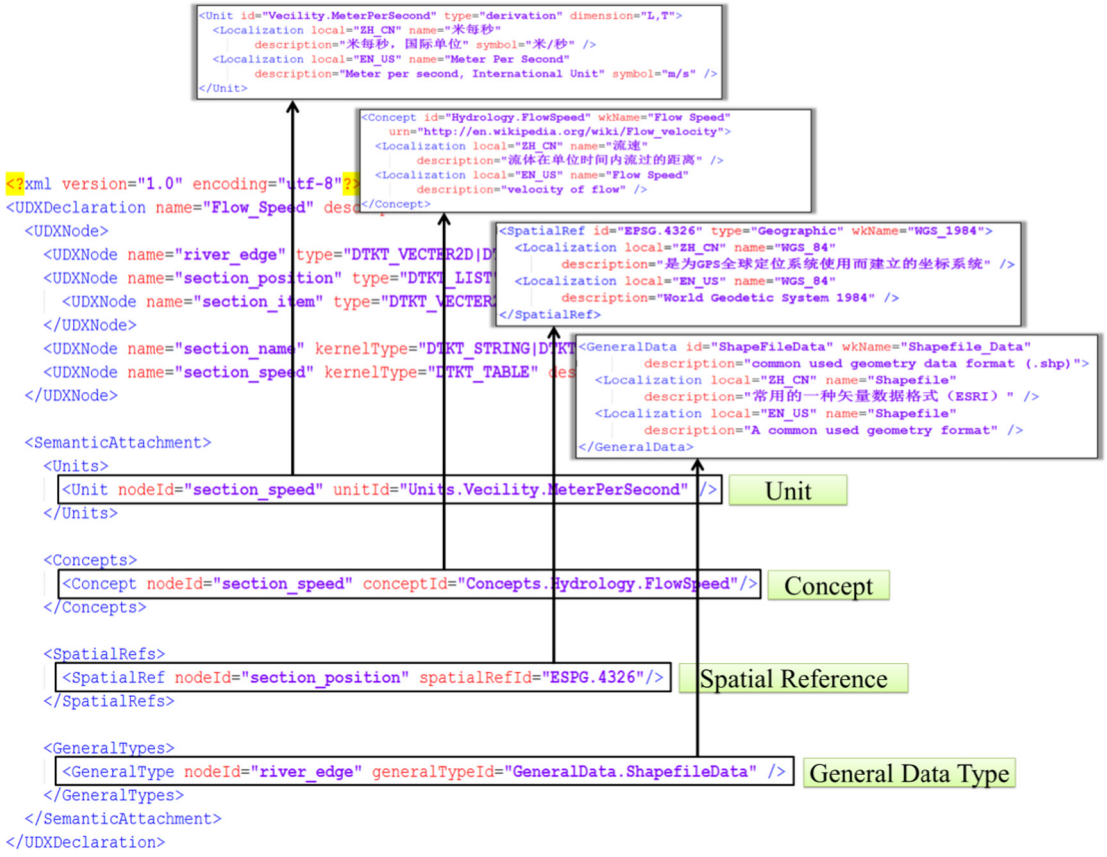
Combined with the UDX model, the execution order and data requirements of geo-analysis models can be represented by the SM-ER model. In addition, the SM-ER model can clearly reflect the execution process and can benefit from collaborative interactions between the models and the external environment. Based on the UDX model and the SM-ER model, the model execution interface is designed as shown in Fig. 9. The execution of the geo-analysis model is described by *IExecutionBehavior*, which includes *DataDeclarationCollection* (all related model data descriptions from the UDX Schema) and *ModelExecutionStates* (describing the execution behaviour of models). Through the *IExecutionBehavior* interface, geo-analysis models can be encapsulated as the interactive "state machine" so that model users can obtain the corresponding model data information (with the UDX Schema) in preparation to use the models and take appropriate actions based on the execution states that are provided by the SM-ER model.

As shown in Fig. 9, the following three basic model execution styles, which are derived from the SM-ER model, are designed to help model practitioners more easily perform model encapsulation: state simulation style, simple calculation style, and time series style.

(1) State simulation style: Geo-analysis models in the state simulation style are encapsulated as execution processes with multiple states, which conform to the SM-ER model. The state simulation style can provide a flexible approach for describing geo-analysis models. The encapsulated models of this style should implement the *IExecutionBehavior* interface, and the *state* concept in the SM-ER model is implemented as *node*. In addition, the state simulation style model should be constructed using one *start-node*, one *end-node*, and a range of *state-nodes* (0–*n*). Every *node* can be invoked using the service-calling method (request/response command) in the general web service environment; therefore, the encapsulated model can be executed through a series of web-service calling commands.




(a) Semantic information of model data




(b) Example implementation of UDX Schema

**Fig. 7.** Design of UDX Schema and a simple example.

(2) Simple calculation style: Although geo-analysis models can represent complex geo-processes, most existing geo-analysis models are implemented as simple mathematical algorithms. The simple calculation style is designed to more easily encapsulate these relatively simple models. The simple calculation style is a special SM-ER model and contains only one *state-node* with one *RequestData* event and one *ResponseData* event. The execution of the encapsulated

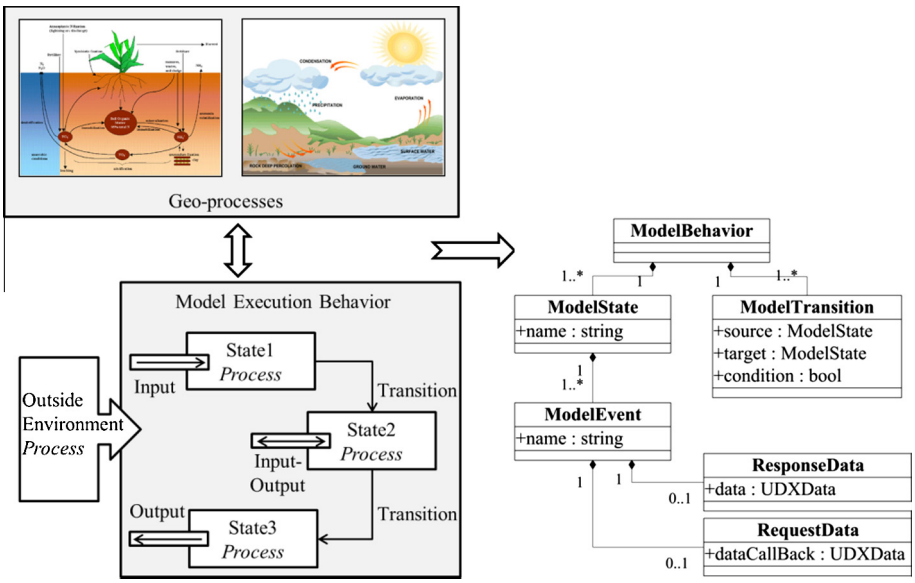



**Fig. 8.** Basic design of the SM-ER model.

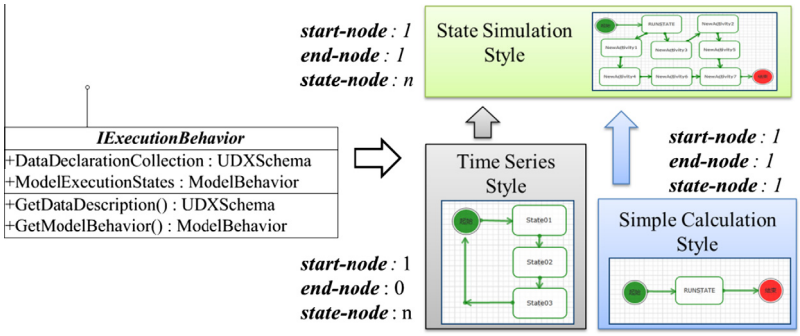



**Fig. 9.** Three basic model execution styles of the SM-ER model.

models in this style is similar to the common web service use method, which involves requesting a service with some parameters and obtaining the response of the execution results.

(3) Time series style: A large number of geo-processes are time dependent and are governed by timelines during the execution of geo-analysis models. The time series style is designed to more easily encapsulate these geo-analysis models. This style is also a special SM-ER model and contains one *start-node* but no *end-node*. The simulation cycle is controlled by the *start-node*; once all state-nodes have been executed, a new cycle of simulation continues if model users provide a *ResponseData* value of YES through the web service.

Regarding the implementation of the model execution interface for encapsulating geo-analysis models, two frequently used and efficient technical methods can be employed: (a) Reprogramming and recompiling the original execution programs. By implementing the UDX model and the SM-ER model, the source codes of the original execution programs of geo-analysis models can be modified and recompiled according to the model-execution interface. (b) Wrapping original execution programs by monitoring the IO operation. For models with no source code, every read/write file operation during the model execution can be monitored using the Hook technology (Wang et al., 2009). The plug-in used for translating original data to the UDX model can be added to original model-execution programs, subsequently forming a new geo-analysis model with a model-execution interface. Both of these methods require effort by programmers. The first method demands a substantial amount of programming; however, this method allows programmers to optimise the data requirements. The second technical method involves less programming but limits the ability to improve the data requirements.

Using the model-execution interface, the heterogeneity of the model data and model-execution behaviour can be reduced, and the encapsulated models can be more conveniently mapped into a web service.

### 3.3. Deployment interface of geo-analysis models

Model execution often involves a massive amount of geographical data processing, the solving of complex equations, and the running of long-sequence simulations, all of which require a large amount of computing resources (NRC, 2010). With the development of Internet-based technologies, especially cloud computing architecture, geo-analysis model services are frequently deployed on distributed computer nodes to fully exploit the computing resources. In addition, the realisation of geo-analysis models depends on diverse hardware architectures, software platforms and programming languages; therefore, heterogeneity occurs at the software level. This deployment is typically very complicated,

and it often requires highly skilled engineers in computer science and web technology.

Generally, in the IT domain, running platform-dependent (e.g., Windows, Linux, UNIX, OSX) and execution-related resources (e.g., configure files, dependent dynamic link libraries) are key elements in distributing software. Moreover, because the software development cycle of geo-analysis models changes over time, several versions of geo-analysis models are developed at various times. Hence, version control is very important for the application of geo-analysis models. Based on all of these characteristics of geo-analysis models, this study designs a flexible and controllable model-deployment interface.

In Fig. 10, the model-deployment interface (*IModelDeployment*) is introduced. *IModelDeployment* consists of the *Entrance* attribute (of the type *ModelEntrance*), the *ConfigureCollection* attribute (a set of *ModelConfigures*), and the *AssemblyCollection* attribute (a set of *ModelAssemblys*). In *ModelEntrance*, *Name* indicates the execution program file name; *Version* provides the version flag and its related web resource link; *Platform* explains the platform dependencies, which are listed in the (extendable) *PlatformEnum*; *BaseDirectory* provides file system information on the execution file; and *ExecutionEntrance* describes the function name used to invoke the model execution programs. In *ModelConfigure*, *key* indicates the model resource type name, and *value* provides the file path information of corresponding resources, such as the software environment (e.g., database software dependencies and system runtime packages), execution-related files (e.g., preloaded configuration files and system demo files), model deployment help files or website URLs. In *ModelAssembly*, *path* indicates the directory of a model-execution-related assembly file, and *name* describes the full name with the extension of this assembly file (e.g., extension of .dll in the WINDOWS environment and .so in LINUX).

Using the model deployment interface, model providers can more clearly describe information on the model deployment in a structured manner. Model users and model service managers can more conveniently prepare model-deployment-related resources and "install" model services on a computation node with the aid of model-deployment interfaces. Thus, the barrier to collaboration between model providers and model users can be reduced.

## 4. Experiment

To demonstrate the capabilities and practicability of the proposed geo-analysis model encapsulation strategy, this study implements the designed model-encapsulation interfaces within the .NET platform and establishes the experimental open web environment under the ASP.NET architecture. Fig. 11 presents the basic process of model sharing and integration using the service-oriented model-encapsulation strategy.

The model providers use four processes: (1) Define the Model. Geo-analysis models are "defined" using the Model Description Interface (which contains the model category system and localisation information). (2) Encapsulate the Model. The original model computation logic is abstracted as the "model core", and the geo-analysis models' programs are encapsulated using the SM-ER model and the UDX model (implementing the Model Execution Interface). (3) Deploy the Model. The Model-Deployment Interface is implemented to help model providers collect and organise all the dependencies of a geo-analysis model.

Through this three-step process, model providers can offer their model resources (represented as the Model Description Documents) to the model-sharing web environment. As shown in the lower right portion of Fig. 11, the Model Description Document is designed to implement model encapsulation results in an integrated manner. The Model Description Document uses XML to organise information regarding the geo-analysis model. The root node is *ModelClass* (a well-known name and a global unique ID) and contains the three following main child nodes: (1) the *AttributeSet* node, a serialisation of the model-description interface, which consists of the *Categories* node (indicates the model category information) and the *Localizations* node (which stores the language-related description information); (2) the *Behaviour* node, a serialisation of the model-execution interface, which consists of the *DatasetDeclaration* node (describes model data) and the *StateGroup* node (describes model execution steps); and (3) the *Runtime* node, a serialisation of the model-deployment interface, which consists of the *Configures* node (the model's runtime resources) and the *Assemblies* node (execution-dependent assemblies).

After submitting the model to the shared web environment via the Portal Website, (1) the model is deployed to the Computation Platform (the Deployment Workspace interprets the deployment information from the Model Description Document), and (2) a corresponding model service item is registered to the Model Resource Library (which can be queried from the Portal Website).

Therefore, from the perspective of model users, (1) the interested geo-analysis model service information can be discovered through the Portal Website, (2) the information for the preparation of running the model of interest can be accessed using the Model Execution Interface, and (3) model users can call and then run model services that are contained in the computation platform via the retransmission of the Portal Website.

Using an example of a typical geo-analysis model's encapsulation process, the proposed model-encapsulation strategy is

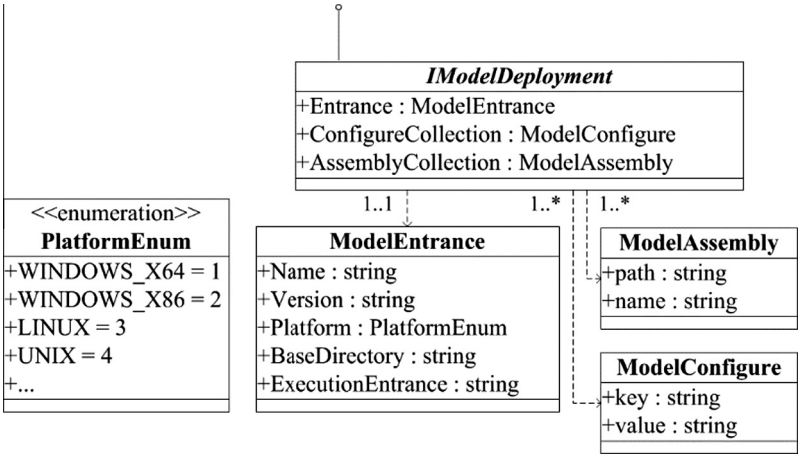



**Fig. 10.** Design of the model-deployment interface.

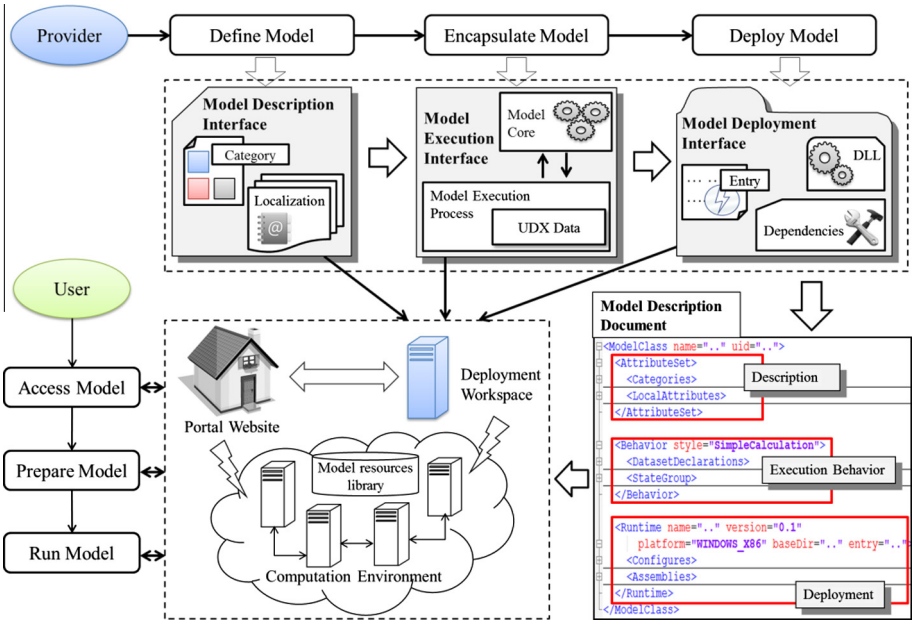



**Fig. 11.** The basic process of encapsulating geo-analysis models.

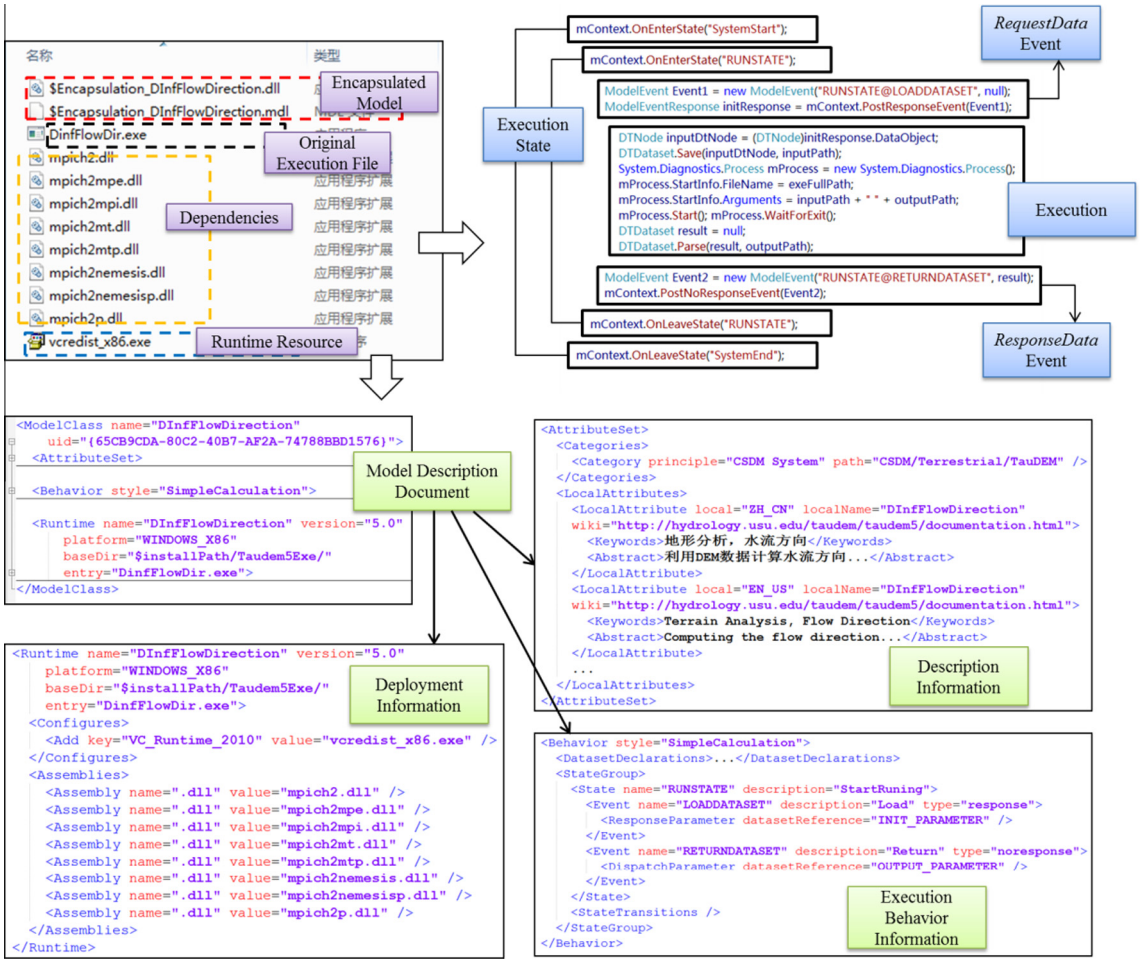



**Fig. 12.** Encapsulation example of the DInfFlowDirection model in TauDEM.

presented in Fig. 12. In this figure, one of the TauDEM models (Terrain Analysis Using Digital Elevation Models, http://hydrology.usu.edu/taudem/taudem5/index.html) is employed to demonstrate the encapsulation process. The top-left portion of Fig. 12 shows the original execution program (Dinfflowdir.exe) as well as its related dependencies (e.g., mpich2.dll and mpich2mpe.dll) and runtime resources (the Visual C++ runtime library, vcredist_x86.exe). A set of software development kits (SDKs) was developed to help modellers implement the SM-ER model to encapsulate their original model resource and to link the execution behaviour with web service calling commands. The basic programming code of the model encapsulation is shown in the top-right of Fig. 12 (using C# style programming code to expound the encapsulation method). The execution of the model should be grouped into execution states (*OnEnterState*/*OnLeaveState* functioning in the *mContext* is the *ModelExecutionContext*). The implementation of *ModelExecutionContext* is dependent on the specific web service architecture. In this study, ASP.Net is employed to contain the *ModelExecutionContext*. For the execution process of the execution state, a *RequestData* Event is invoked to obtain input data. After the execution of the specific model, a *ResponseData* Event is invoked to return the computed result. After encapsulation, the execution process can be considered as entering the *SystemStart* state, entering the *Run* state, invoking a *RequestData* event, executing the computation with input data, returning the result of a *ResponseData* event, and exiting the *Run* State or exiting the *SystemStart* state.

The model description document is constructed using XML, as shown at the bottom of Fig. 12. The name of this model is "DInfF lowDirection", and a GUID is assigned to represent the unique model. (1) The model-description interface is implemented as the *AttributeSet* node, with category description of "CSDM System" and "CSDM/Terrestrial/TauDEM" and localisation attributes in Chinese and English. (2) The model-execution interface is implemented as the *Behaviour* node, and the *DataDeclarations* node contains the UDX Schemas that indicate how to prepare the input data (as GeneralData.ShapefileData for this model) and define the output data (as GeneralData.GeoTiffData for this model). In addition, a "*RUNSTATE*" node that contains one load data event and one return data event is used to wrap the original execution process. (3) The model-deployment interface is implemented as the *Runtime* node, with a version number of "5.0", a platform dependency of "Windows_X86", and other deployment-related resources that are given in the *Configures* and *Assemblies* nodes.

In Fig. 13, the GUI tool used to implement the geo-analysis model-encapsulation process is presented. The *model description panel* is developed to help model providers describe model category system information and model localisation information. The *model behaviour definition panel* is developed to help model providers describe the execution states of the model. The definition of these model execution states should track the real execution process of the corresponding encapsulated model program. Finally, the *model data definition panel* is developed to help model providers define and convert original model data used within the model execution states.

Based on the implementation results shown in Figs. 11–13, a variety of Web Service technologies can be employed to publish model services. This study uses the WPS standard as an example to transmit data between clients and servers. As shown in Fig. 14, the *DescribeProcess* request contains information regarding the model declaration document information that can be obtained using an external link, and model users can access this information by visiting a new URL. Similarly, the *DataInputs* and *ProcessOutputs* nodes describe the model data using external links that contain the

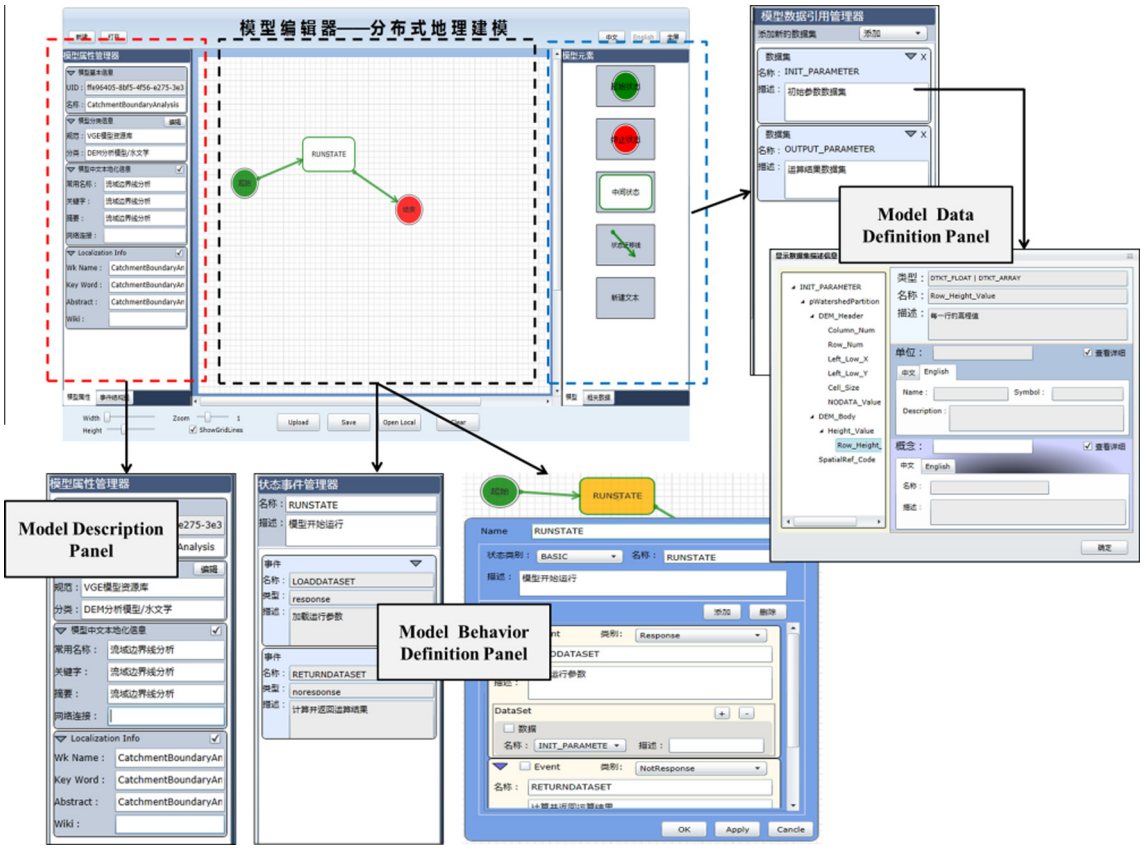
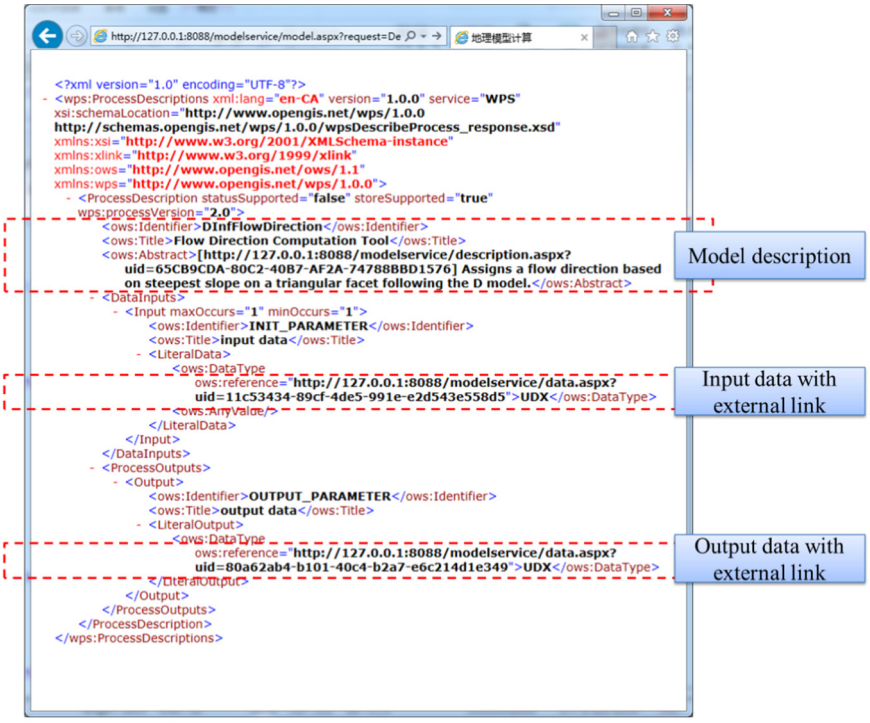
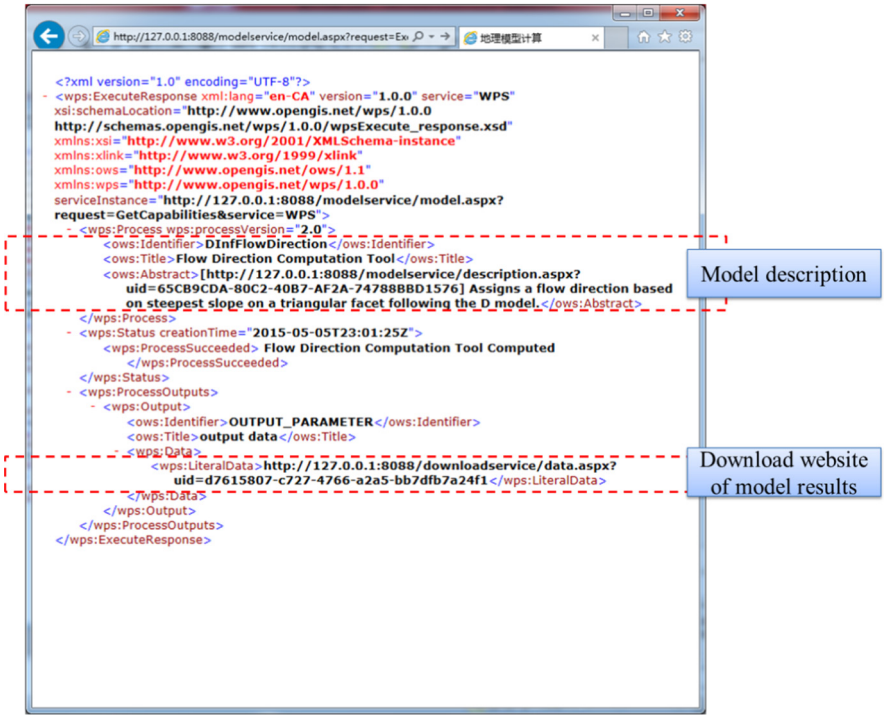



**Fig. 13.** The implementation of the model encapsulation interfaces.




(a) *DescribeProcess* request of the DInfFlowDirection model




(b) *Execute* request of the DInfFlowDirection model

**Fig. 14.** Using the WPS standard to publish a model service.

UDX Schema. In this manner, the original organisation of the WPS standards is the same, and information regarding the model service is extended. Regarding the *Execution* operation, the input UDX data should be uploaded to the data server, and the original XML file (posted to the WPS server) uses the URL of the UDX data as Input-Data. This study developed a WPS server program to interpret web requests and conduct model execution. The computation results are provided to the user clients as downloadable files, and the URLs are contained in the WPS XML output document.

According to the above analysis, the project group designed and developed a prototype system of sharing, integrating and executing geo-analysis models in an open web environment. Fig. 15 shows some of the geo-analysis models in the model library built by encapsulating typical models from several disciplines. In

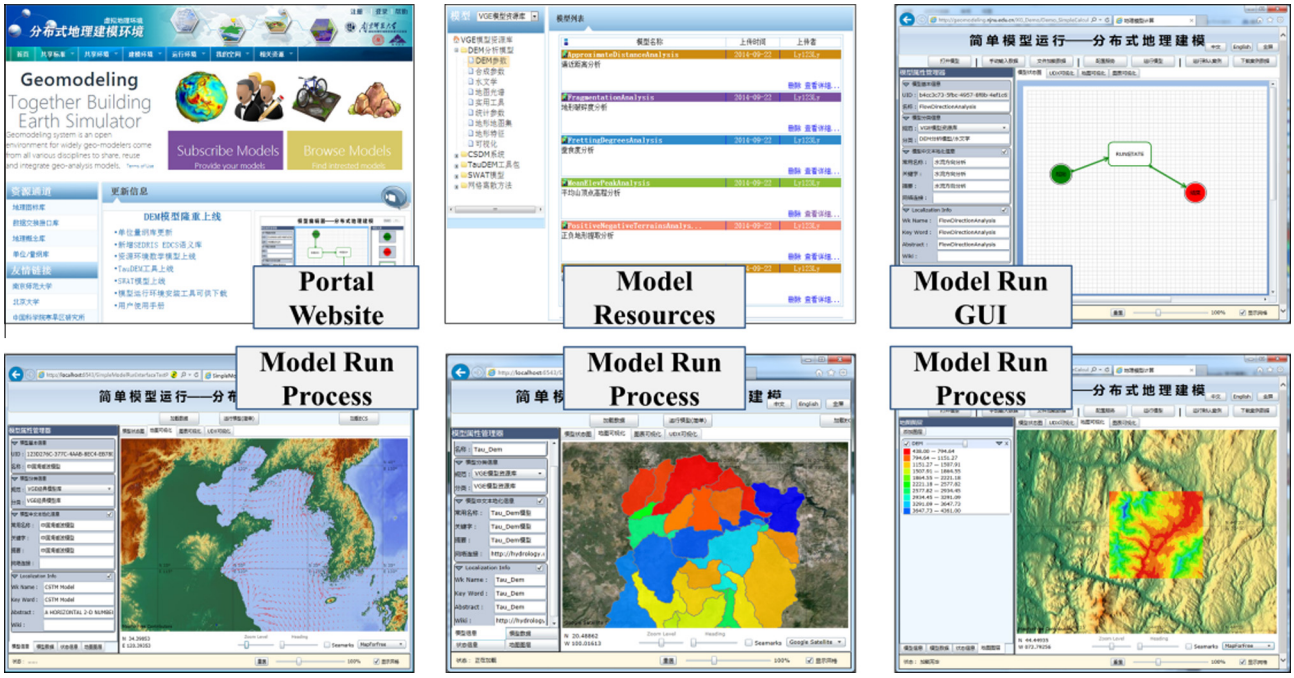



**Fig. 15.** Prototype system of sharing and integrating geo-analysis models.

addition, geo-analysis models from several disciplines were encapsulated and deployed in an array of computing servers, and a portal website was constructed to manage all the model services. Model users can call these model services by performing a request using the model GUID and model service URL.

## 5. Conclusions and future research

The purpose of this study was to simplify model encapsulation, which is important in the reuse, sharing and integration of geo-analysis models. Using the proposed model-encapsulation strategy, heterogeneous geo-analysis models can be described in a common manner. With the proposed model-description interface, model-execution interface, and model-deployment interface, geo-analysis models and model-related resources can be described, executed and deployed more clearly, thus making it more convenient for modellers to reuse, share and integrate heterogeneous geo-analysis models.

However, geo-modelling and geo-model integration research is ongoing, and the following research topics should be addressed in the future:

(1) A more complete and automated strategy to deploy model resources. The technology and architecture of geo-analysis models vary between different model developers. The proposed model-encapsulation strategy can provide a clearer and more structured method to encapsulate these heterogeneous models; however, the deployment of model resources still requires a substantial amount of manual effort. By interpreting the deployment information, an automated or semi-automated method of deploying model services is required to satisfy the demands of building an extendable model library in a web environment.

(2) Designing the UDX data operation method and building the data processing library based on the UDX model. After the encapsulation of geo-analysis models, input and output data of the model execution can be formed in a common manner. However, model data pre-processing is still necessary and essential for using and integrating geo-analysis models, especially for interdisciplinary modelling, and it often demands a substantial amount of programming efforts. Building an extensive model data-processing library can reduce the effort required of model users. Moreover, UDX schemas require further research and development, whereby an effective mapping library of existing data formats to UDX schemas should be established so that model practitioners can more efficiently encapsulate existing geo-analysis models.

(3) A set of utilities must be developed to help modellers more easily encapsulate geo-analysis models. Using the IO hook technology, model-encapsulation methods can be more efficiently programmed and recompiled. In addition, a programming template should be designed and developed to help modellers more effectively write the encapsulation program code.

(4) More completed applications that use different web transmission methods (HTTP, P2P and WebSocket) to construct modelling environments or geo-processing systems should be designed and developed. When implementing model services using the WPS standard, it is simple to verify the capabilities of the model-encapsulation strategy. In addition, the WPS standard can be extended to support more comprehensive descriptions of geo-analysis models. Based on cloud computing, the encapsulated results of geo-analysis models can be deployed using different computation servers, and the web transmission method is related to specific cloud computing platforms.

## Acknowledgements

We appreciate the detailed suggestions and comments from the editor and the anonymous reviewers. The work described in this article involves many geo-analysis models and was supported by the National Basic Research Program of China (973 Program, Grant No. 2015CB954102), the National Natural Science Foundation of China (Grant Nos. 41471317 and 41371424), the Program of

Natural Science Research of Jiangsu Higher Education Institutions of China (Grant No. 13KJB170008), and the Priority Academic Program Development of Jiangsu Higher Education Institutions (Grant No. 164320H116), the National Science and Technology Support Plan Project (Grant No. 2012BAH35B02).

## References


Albanesi, M.G., Albanesi, R., 2014. A decision-making support system for land use estimation based on a new anthropentropy predictive model for environmental preservation-theory, model, and web-based implementation. Int. J. Adv. Intell. Syst. 7 (1 and 2), 85–102.

Alonso, G., Hagen, C., 1997. Geo-Opera: workflow concepts for spatial processes. In: Advances in Spatial Databases. Springer, Berlin Heidelberg, pp. 238–258.

Argent, R.M., 2004. An overview of model integration for environmental applications—components, frameworks and semantics. Environ. Model. Softw. 19 (3), 219–234.

Argent, R.M., Voinov, A., Maxwell, T., Cuddy, S.M., Rahman, J.M., Seaton, S., Vertessy, R.A., Braddock, R.D., 2006. Comparing modelling frameworks – a workshop approach. Environ. Model. Softw. 21 (7), 895–910.

Arnold, J.G., Srinivasan, R., Muttiah, R.S., Williams, J.R., 1998. Large area hydrologic modeling and assessment part I: Model development 1. 10.1111/j.1752-1688.1998.tb05961.x.

Basnyat, P., Teeter, L.D., Lockaby, B.G., Flynn, K.M., 2000. The use of remote sensing and GIS in watershed level analyses of non-point source pollution problems. For. Ecol. Manage. 128 (1), 65–73.

Bergner, J., 2007. A model category structure on the category of simplicial categories. Trans. Am. Math. Soc. 359 (5), 2043–2058.

Blind, M., Gregersen, J.B., 2005. Towards an open modelling interface (OpenMI) the HarmonIT project. Adv. Geosci. 4 (4), 69–74.

Britz, W., Domínguez, I.P., Heckelei, T., 2010. A comparison of CAPRI and SEAMLESS-IF as integrated modelling systems. In: Environmental and Agricultural Modelling. Springer, Netherlands, pp. 257–274.

Chen, M., Lin, H., Lu, G., He, L., Wen, Y., 2011. A spatial-temporal framework for historical and cultural research on China. Appl. Geogr. 31, 1059–1074.

Chen, Z., Lin, H., Chen, M., Liu, D., Bao, Y., Ding, Y., 2014. A framework for sharing and integrating remote sensing and GIS models based on Web service. Sci. World J. 2014.

Deelman, E., Gannon, D., Shields, M., Taylor, I., 2009. Workflows and e-Science: an overview of workflow system features and capabilities. Future Gener. Comput. Syst. 25 (5), 528–540.

Dennis, J.M., Vertenstein, M., Worley, P.H., Mirin, A.A., Craig, A.P., Jacob, R., Mickelson, S., 2012. Computational performance of ultra-high-resolution capability in the Community Earth System Model. Int. J. High Perform. Comput. Appl. 26 (1), 5–16.

Dickinson, R.E., Oleson, K.W., Bonan, G., Hoffman, F., Thornton, P., Vertenstein, M., Yang, Z., Zeng, X., 2006. The Community Land Model and its climate statistics as a component of the Community Climate System Model. J. Clim. 19 (11), 2302–2324.

Erl, T., 2008. Soa: Principles of Service Design, vol. 1. Prentice Hall, Upper Saddle River.

Geller, G.N., Turner, W., 2007. The model web: a concept for ecological forecasting. In: Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International, pp. 2469–2472 (IEEE).

Goodchild, M.F., Steyaert, L.T., Parks, B.O. (Eds.), 1996. GIS and Environmental Modeling: Progress and Research Issues. John Wiley & Sons.

Granell, C., Schade, S., Ostländer, N., 2013. Seeing the forest through the trees: a review of integrated environmental modelling tools. Comput. Environ. Urban Syst. 41, 136–150.

Hillyer, C., Bolte, J., van Evert, F., Lamaker, A., 2003. The ModCom modular simulation system. Eur. J. Agron. 18 (3), 333–343.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughesm, A., 2013. Integrated environmental modeling: a vision and roadmap for the future. Environ. Model. Softw. 39, 3–23.

Lei, X., Wang, Y., Liao, W., Jiang, Y., Tian, Y., Wang, H., 2011. Development of efficient and cost-effective distributed hydrological modeling tool MWEasyDHM based on open-source MapWindow GIS. Comput. Geosci. 37 (9), 1476–1489.

Li, W., Goodchild, M.F., Robert, R., 2014. Towards geospatial semantic search: exploiting latent semantic relations in geospatial data. Int. J. Digit. Earth 7 (1), 17–37.

Li, Y., Gong, J., Zhu, J., Song, Y., Hu, Y., Ye, L., 2013. Spatiotemporal simulation and risk analysis of dam-break flooding based on cellular automata. Int. J. Geogr. Inform. Sci. 27 (10), 2043–2059.

Li, Y., Gong, J., Liu, H., Zhu, J., Song, Y., Liang, J., 2015. Real-time flood simulations using CA model driven by dynamic observation data. Int. J. Geogr. Inform. Sci. http://dx.doi.org/10.1080/13658816.2014.977292.

Lin, H., Batty, M., Jørgensen, S.E., Fu, B., Konecny, M., Voinov, A., Torrens, P., Lu, G., Zhu, A., Wilson, J.P., Gong, J., Kolditz, O., Bandrova, T., Chen, M., 2015. Virtual environments begin to embrace process-based geographic analysis. Trans. GIS 19 (4), 493–498.

Lin, H., Chen, M., Lu, G., Zhu, Q., Gong, J., You, X., Wen, Y., Xu, B., Hu, M., 2013a. Virtual Geographic Environments (VGEs): a new generation of geographic analysis tool. Earth Sci. Rev. 126, 74–84.

Lin, H., Chen, M., Lu, G., 2013b. Virtual geographic environment: a workspace for computer-aided geographic experiments. Ann. Assoc. Am. Geogr. 103 (3), 465–482.

Malaverri, J.E., Medeiros, C.B., Lamparelli, R.C., 2012. A provenance approach to assess the quality of geospatial data. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 2043–2044 (ACM).

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

Maxwell, T., Costanza, R., 1997b. A language for modular spatio-temporal simulation. Ecol. Model. 103 (2), 105–113.

McColl, C., Aggett, G., 2007. Land-use forecasting and hydrologic model integration for improved land-use decision support. J. Environ. Manage. 84 (4), 494–512.

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

Nativi, S., Mazzetti, P., Geller, G.N., 2013. Environmental model access and interoperability: The GEO Model Web initiative. Environ. Model. Softw. 39, 214–228.

National Research Council (US). Committee on Modeling, Simulation, and Games, 2010. The Rise of Games and High-performance Computing for Modeling and Simulation. National Academies Press.

OGC Provenance Engineering Report, 2014 <https://portal.opengeospatial.org/files/?artifact_id=58967> (accessed May, 2015).

Parsons, M.A., 2011. Making data useful for modelers to understand complex Earth systems. Earth Sci. Inf. 4 (4), 197–223.

Peckham, S.D., Hutton, E., 2009. Componentizing, standardizing and visualizing: how CSDMS is building a new system for integrated modeling from open-source tools and standards. In: AGU Fall Meeting Abstracts, vol. 1, p. 1045.

Pullar, D., Springer, D., 2000. Towards integrating GIS and catchment models. Environ. Model. Softw. 15 (5), 451–459.

Rodriguez, L.B., Cello, P.A., Vionnet, C.A., anf Goodrich, D., 2008. Fully conservative coupling of HEC-RAS with MODFLOW to simulate stream–aquifer interactions in a drainage basin. J. Hydrol. 353 (1), 129–142.

Sancho-Jiménez, G., Béjar, R., Latre, M.A., Muro-Medrano, P.R., 2008. A method to derivate SOAP interfaces and WSDL metadata from the OGC web processing service mandatory interfaces. In: Advances in Conceptual Modeling–Challenges and Opportunities. Springer, Berlin Heidelberg, pp. 375–384.

Serreze, M.C., 2011. Climate change: rethinking the sea-ice tipping point. Nature 471 (7336), 47–48.

Succar, B., 2009. Building information modelling framework: a research and delivery foundation for industry stakeholders. Automat. Constr. 18 (3), 357–375.

Sudo, K., Takemura, T., Klimont, Z., Kurokawa, J., Akimoto, H., 2013. Global modeling and projection of short-lived climate pollutants in an earth system model. In: AGU Fall Meeting Abstracts, vol. 1, p. 08.

Tilmes, C., Fleig, A.J., 2008. Provenance tracking in an earth science data processing system. In: Provenance and Annotation of Data and Processes. Springer, Berlin Heidelberg, pp. 221–228.

Todorova, A., Syrakov, D., Gadjhev, G., Georgiev, G., Ganev, K.G., Prodanova, M., Miloshev, N., Spiridonov, V., Bogatchev, A., Slavov, K., Slavov, K., 2010. Grid computing for atmospheric composition studies in Bulgaria. Earth Sci. Inf. 3 (4), 259–282.

Valcke, S., Guilyardi, E., Larsson, C., 2006. PRISM and ENES: a European approach to Earth system modelling. Concurr. Comput.: Pract. Exp. 18 (2), 247–262.

Wang, S., Padmanabhan, A., Myers, J.D., Tang, W., Liu, Y., 2008. Towards provenance-aware geographic information systems. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 70 (ACM).

Wang, Y., Shen, Y., Pan, J., 2009. Usage control based on windows kernel hook. In: International Conference on Information and Multimedia Technology, 2009. ICIMT'09, pp. 264–267 (IEEE).

Wen, Y., Chen, M., Lu, G., Lin, H., He, L., Yue, S., 2013. Prototyping an open environment for sharing geographical analysis models on cloud computing platform. Int. J. Digit. Earth 6 (4), 356–382.

Xu, X., Huang, G., Zhan, H., Qu, Z., Huang, Q., 2012. Integration of SWAP and MODFLOW-2000 for modeling groundwater dynamics in shallow water table areas. J. Hydrol. 412, 170–181.

Yin, L., Zhu, J., Zhang, X., Li, Y., Wang, J., Zhang, H., Yang, X., 2015. Visual analysis and simulation of dam-break flood spatiotemporal process in a network environment. Environ. Earth Sci. http://dx.doi.org/10.1007/s12665-015-4418-3.

Yue, P., Gong, J., Di, L., 2010. Augmenting geospatial data provenance through metadata tracking in geospatial service chaining. Comput. Geosci. 36 (3), 270–281.

Yue, S., Wen, Y., Chen, M., Lu, G., Hu, D., Zhang, F., 2015. A data description model for reusing, sharing and integrating geo-analysis models. Environ. Earth Sci. 74 (10), 7081–7099.

Zhang, C., Lin, H., Chen, M., Li, R., Zeng, Z., 2014. Scale compatibility analysis in geographic process research: a case study of a meteorological simulation in Hong Kong. Appl. Geogr. 52, 135–143.

Zhang, C., Chen, M., Ding, Y., Li, R., Lin, H., 2015a. A multiscale virtual geographic environments system for studying air quality process, with a case study of $SO_2$ concentration simulation in Hong Kong. Appl. Geogr. 63, 326–336.

Zhang, C., Chen, M., Jørgensen, S.E., Lin, H., Li, R., 2015b. What's going on about geo-process modeling in virtual geographic environments (VGEs). Ecol. Model. http://dx.doi.org/10.1016/j.ecolmodel.2015.04.023.

Zhu, J., Zhang, H., Chen, M., Xu, Z., Qi, H., Yin, Y.Z., Wang, J.H., Hu, Y., 2015. A procedural modelling method for virtual high-speed railway scenes based on model combination and spatial semantic constraint. Int. J. Geogr. Inform. Sci. http://dx.doi.org/10.1080/13658816.2015.1008493.