International Journal of Digital Earth

ISSN: 1753-8947 (Print) 1753-8955 (Online) Journal homepage: http://www.tandfonline.com/loi/tjde20

A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment

Yongning Wen, Min Chen, Songshan Yue, Peibei Zheng, Guoqiang Peng & Guonian Lu

To cite this article: Yongning Wen, Min Chen, Songshan Yue, Peibei Zheng, Guoqiang Peng & Guonian Lu (2017) A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment, International Journal of Digital Earth, 10:4, 405-425, DOI: 10.1080/17538947.2015.1131340

To link to this article: http://dx.doi.org/10.1080/17538947.2015.1131340

Published online: 26 Jan 2016.

INTERNATIONAL JOURNAL OF DIGITAL EARTH, 2017
VOL. 10, NO. 4, 405–425
http://dx.doi.org/10.1080/17538947.2015.1131340

A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment

Yongning Wen[a,b,c], Min Chen[a,b,c,d], Songshan Yue[a,b,c], Peibei Zheng[a,b,c], Guoqiang Peng[a,b,c] and Guonian Lu[a,b,c]

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

ABSTRACT

Geo-analysis models can be shared and reused via model-services to support more effective responses to risks and help to build a sustainable world. The deployment of model-services typically requires significant effort, primarily because of the complexity and disciplinary specifics of geo-analysis models. Various modelling participants engage in the collaborative modelling process: geo-analysis model resources are provided by model providers, computational resources are provided by computational resource providers, and the published model-services are accessed by model users. This paper primarily focuses on model-service deployment, with the basic goal of providing a collaboration-oriented method for modelling participants to conveniently work together and make full use of modelling and computational resources across an open web environment. For model resource providers, a model-deployment description method is studied to help build model-deployment packages; for computational resource providers, a computational resource description method is studied to help build model-service containers and connectors. An experimental system for sharing and reusing geo-analysis models is built to verify the capability and feasibility of the proposed methods. Through this strategy, modellers from dispersed regions can work together more easily, thus providing dynamic and reliable geospatial information for Future Earth studies.

ARTICLE HISTORY
Received 28 August 2015
Accepted 9 December 2015

KEYWORDS
Model service deployment; model sharing; modelling environment

1. Introduction

Along with Future Earth studies that seek to integrate and develop knowledge from all related research disciplines and further support more effective responses to risks while building a sustainable world (Griggs et al. 2013; Chen, Ban, and Li 2014; Dobbs, Kendal, and Nitschke 2014), there is an increasing demand for the ability to integrate multi-disciplinary knowledge and process complex geospatial information (Parsons 2011; Li, Xiong, and Ou 2011; Butt and Li 2012; Veenendaal et al. 2014). Geo-analysis models that come from various research fields, including both social dimension (e.g. Chen et al. 2013a; Torrens and McDaniel 2013; Luo and MacEachren 2014, Luo et al. 2014; Torrens 2015) and physical dimension (e.g. Gutierrez et al. 2013; Jørgensen 2015; Li et al. 2015), can be employed to solve complex geo-problems, support comprehensive decisions,

CONTACT Songshan Yue yss123yss@126.com

and process more dynamic and synthetic geospatial information for Future Earth studies (Basnyat et al. 2000; Claussen et al. 2002; Brovelli et al. 2013; Lin et al. 2015). In this context, the Digital Earth concept was initially proposed to assist researchers in integrating, utilizing, and sharing data and information resources; currently, however, this concept is entering a new stage that calls for predicting the future state of the Earth by employing and integrating various mechanism processes and models (Craglia et al. 2008; Goodchild 2008, 2012; Guo, Liu, and Zhu 2010; Goodchild et al. 2012; Skidmore and Woodgate 2012; Chen et al. 2015), for which geo-analysis models are a very important tool (Zhang et al. 2015).

Geo-analysis models are developed by research individuals or groups scattered all over the world, and the execution of geo-analysis models often involves substantial computational power and significant amounts of data (Dickinson et al. 2006; Zhang, Zhao, and Li 2010; Hofer 2014; Müller 2015). Collecting these model (and model-related) resources in a centralized mode can reduce the difficulty of sharing knowledge to some extent; however, the collaboration of dispersed researchers is limited (Reichardt 2010; Arciniegas, Janssen, and Rietveld 2013). In response, the development of web technology in sharing, reusing, and integrating geo-analysis models in a web environment has been widely accepted and approved as an efficient way to help modellers work collaboratively (Chen et al. 2013b; Chang and Li 2013; Laniak et al. 2013).

The continual evolution of the Internet and web technology has largely promoted the efficient management, sharing, and computation methods of geo-analysis model resources (Granell, Díaz, and Gould 2010; Yang et al. 2011; Zhao, Foerster, and Yue 2012). 'Distributed computing' has been widely used in processing geospatial data over the web; and various distributed computing platforms, such as grids, clusters, and clouds, are available (Foster et al. 2008; Todorova et al. 2010). Grid computing and cluster computing primarily focus on intensive resource sharing and innovative applications (Shi, Shortridge, and Bartholic 2002; Chen et al. 2009) and are widely used to conduct high-performance computing (Thakker, Abdul, and Potdar 2015); cloud computing has been approved as one of the most promising approaches to make full use of dispersed resources and provide powerful computation capability (Tsai, Sun, and Balasooriya 2010; Wen et al. 2013). Characterized by transparency, openness, and extendable, adaptable, and alternative services, cloud computing provides the possibility of dynamically sharing and integrating various geo-analysis models (Yang, Xu, and Nebert 2013; Evangelidis et al. 2014). These web-service architectures can facilitate the efficient sharing of distributed model resources, and an open web environment for sharing, reusing, and integrating geo-analysis models and model-related resources can be built based on a hybrid of different computing architectures (Khan et al. 2012; Yue et al. 2013). Extensive efforts have been applied to model-service and geo-processing methods, including the evaluation of the scalability feature of a Web Processing Service (WPS) implemented in the cloud (Yue et al. 2013), the design, and testing of integrating cloud computing with spatial data infrastructures (SDIs) (Steiniger and Hunter 2012), the expansion of combining local and public IT-infrastructure to meet quality of service (QoS) requirements (Baranski et al. 2011; Li et al. 2012), and research on placing geospatial services in the cloud (Krämer and Senner 2015). Modellers can more conveniently communicate with each other through this open web environment.

Since the proposal of service-oriented architecture (SOA), a range of research and technical approaches have been studied to allow for interactions between 'service providers' (servers) and 'service consumers' (clients). According to SOA, geo-analysis models and data are offered as services that are available via web communication interfaces or standards. The Web Service Description Language (WSDL) and Open Geospatial Consortium, Web Processing Service (OGC WPS) standards are two notable and continually updated web communication approaches used to build modelling services (also called geo-processing services) (Han et al. 2015; Thakker, Abdul, and Potdar 2015). Both WSDL-based and WPS-based modelling services focus on the provision of capabilities, such as the ability to access data and invoke data processing computations (Giuliani et al. 2012). With regard to sharing and integrating geo-analysis models through the web, many researchers have studied methods to transition WPS services (Schaeffer 2008), to search for and acquire

geospatial data (Evangelidis et al. 2014), to develop a geo-processing workflow (Yang, Xie, and Xu 2014), to compose and orchestrate different geo-web services (Di Giovanni et al. 2014; Wu et al. 2014), and to discover and select model services (Matheus 2013; Wang et al. 2014; Lartigau et al. 2015). Although different model services can be coupled to conduct distribution computations (e.g. web service chains and orchestrations), the lack of user control and interactivity between model services and model providers/users still limit reusability and interoperability when solving complex geo-problems and simulating synthetic Earth environments in an open web environment (Laniak et al. 2013; Lin, Chen, and Lu 2013a, 2013b). The available model services are published by research groups or enterprises, but it is difficult for volunteers to contribute and publish their models as a model service (Miller and Goodchild 2014; Hou et al. 2015).

Among these problems, deploying various geo-analysis models in different distributed compute nodes is one of the difficulties that significantly influence the sharing and integrating of model resources. When modellers attempt to collaboratively work to solve complex problems, the deployment of geo-analysis model services typically requires excessive effort due to the complexity and disciplinary specifics of geo-analysis models (the execution of a model can require specific hardware conditions and software environments). In addition, to fully utilize distributed computational resources,

different modelling participants must be involved in the collaborative modelling process: (1) model providers can provide various geo-analysis model resources but may not have the necessary computational resources to publish their model services; (2) computational resource providers can provide different computational resources and help with publishing model services; and (3) based on the collaboration of model providers and computational resource providers, models must be deployed in appropriate compute nodes where model users can access the published model services. Furthermore, in some cases, model users may wish to transfer their interested model services to their own computers (local or worldwide); thus, information on how to deploy these model services is also necessary.

In order for modellers to successfully deploy a model service, it is important that the modellers understand the model requirements (e.g. hardware/software dependencies and internally referenced data). In addition, computational resources also relate to the deployment of a variety of model services. In an open web environment, a range of diverse compute nodes can be provided by individuals or organizations; modellers must discover and select appropriate compute nodes according to the demands of a specific model service and then deploy these services. This study primarily focuses on reducing the difficulties of deploying a variety of geo-analysis model services that significantly affect and hinder the collaboration of modellers to conduct integrated modelling research. The basic goal of this deployment strategy is to provide a collaboration-oriented method for modellers to more conveniently work together and make full use of model and computational resources across an open web environment. Therefore, this paper studies the methods for describing model-deployment information and computational resource information. Based on this information, a collaborative strategy is designed (1) for model providers to describe the deployment information, (2) for computation providers to describe the computational information, and (3) for modellers to deploy model resources in distributed compute nodes and coordinate the organization of various model services.

The basic concept behind the collaboration-oriented deployment strategy is introduced in Section 2. In Section 3, the detailed model-deployment information description method and computational information description method are explained and the computational registration and deployment management architecture is also introduced. Section 4 presents the realization of a prototype system and related experiments. Finally, the conclusions and a discussion of the study are presented in Section 5.

2. Basic concept

Based on the above analysis, model services should be published in a compute node (also called a web node, cloud node, or server) so that model users can recall and invoke them to conduct distributed

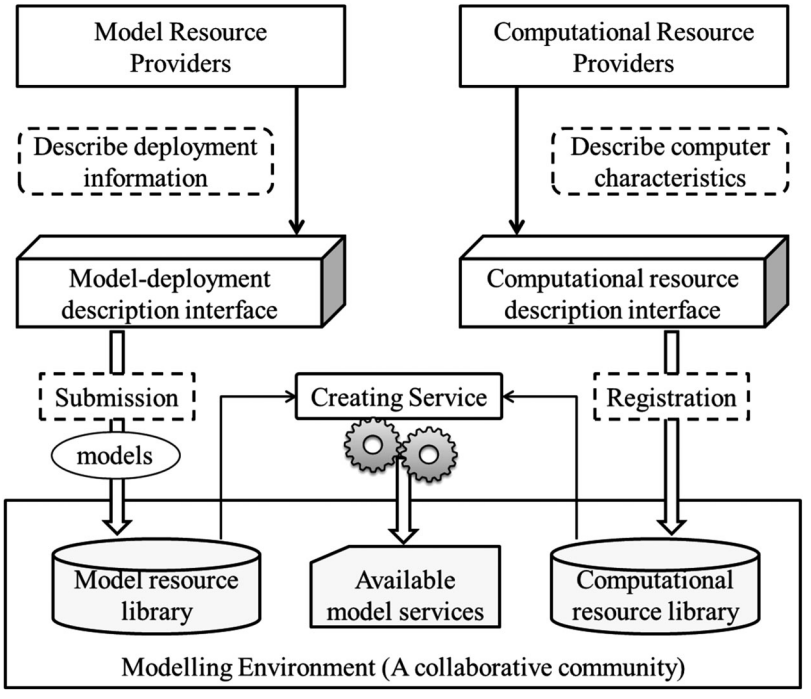


Figure 1. Basic concept of the model-deployment strategy.

computation. As seen in Figure 1, the model-deployment strategy consists of two main processes: (1) model resource providers describe the ready-to-be-offered model and submit this to the model resource library as formed in the modelling environment; and (2) computational resource providers sufficiently describe the ready-to-be-offered compute node and register it into the computational resource library as organized in the modelling environment. Different model services can be created and published.

through these two libraries, and general modellers can discover relevant model resources and access available model services. The two methods are designed to help modelling participants more conveniently contribute their model resources or computational resources into the modelling environment and build reusable and available model services.

The model-deployment description interface is primarily designed to describe the deployment information of a geo-analysis model. Because the realization of geo-analysis models depends on diverse hardware architectures, software platforms, and programming runtimes, the organization of this information should be flexible and extendable. Both description and packaging methods are required to help model providers more easily encapsulate original models and support more automatic service deployment.

The computation resource description interface must include methods that describe the conditions (deployment-related) of a ready-to-be-offered compute node. Model services should be published using certain service containers, such as Internet Information Service (IIS, Microsoft), Apache (Oracle), or other applications. Once a model service has been published, the communication methods between clients and the server are assigned [e.g. by using a SOAP-based (Simple Object Access Protocol) method or REST-based (Representation State Transmission) method]. Thus, the methods used to organize the service container information, service communication information, and computer conditions are included in the computational resource description interface.

Based on the model-deployment description interface and computational resource description interface, model resources can be deployed in an appropriate compute node and thus create available

model services. The work involved in creating the service process must be validated, meaning that the computational resource information should be compared with the model-deployment information. With this interface-based design, a structural communication channel can be built across various modellers and resource providers.

3. Deployment strategy of geo-analysis model resources

3.1 Model-deployment description method

Geo-analysis models are developed by various modellers, and the sharing and integrating of geo-analysis models largely depends on fully understanding the deployment-related information. In this case, understanding refers to both human-readable (so model users more completely understand a model and can conduct more sophisticated applications) and machine-readable (to support developing more automatic deployment tools) information. Based on the description of the model-deployment information, deployment packages should be built and uploaded to compute nodes. Therefore, methods to organize the deployment information of geo-analysis models and build deployment packages are necessary.

3.1.1 Deployment information of geo-analysis models

Geo-analysis models can generally be represented as mathematical models and implemented as executable programs with specific parameters and runtime platform requirements (Granell, Schade, and Ostländer 2013). For service users, one of the major advantages of 'web services' is reducing the use of certain software that requires deploying a range of execution files in their own computers; for service providers (service managers), this deployment work is necessary at the server end. Three types of model-deployment-related information are shown in Figure 2.

Computer information primarily contains a description of hardware conditions. The execution of geo-analysis models typically requires sufficient hardware conditions, such as memory size, disk storage size, and CPU/GPU type. Some models that employ multi-core parallel computing technology need to be installed on a machine with a multi-core CPU, and some models that use GPU to process computation require a machine with appropriate types of GPU. These hardware requirements must be verified before deploying some specific geo-analysis models in certain machines.

Software information primarily describes the required software environment of the geo-analysis models. The implementation of geo-analysis models is typically characterized by the individual inclinations and habits of the developer. For example, certain model developers may use a single file to store the input data of a model, whereas others may employ multiple files to organize the input data;

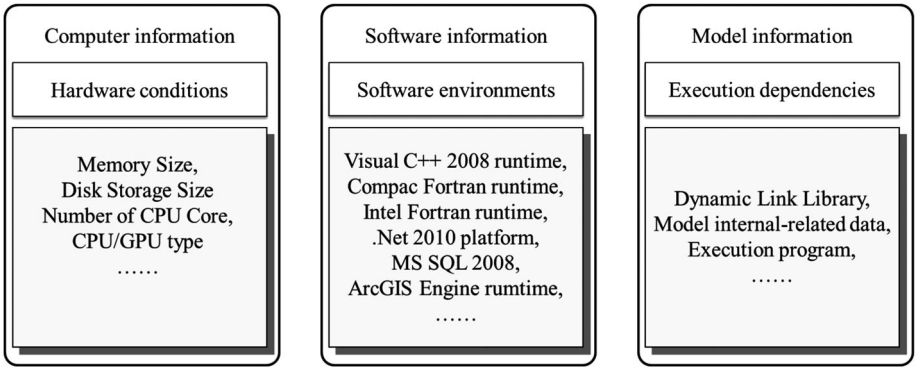


Figure 2. Model-deployment-related information.

certain model developers prefer executing models on the Linux platform, whereas others lean towards executing models on the Windows platform. Thus, various technology approaches can be employed to help develop models, such as different operating systems, program compiling environments, and other open-source or commercial software development kits. The deployed geo-analysis models cannot be executed without the dependent software.

Model information primarily indicates the content of a geo-analysis model. Along with the evolution of software engineering, a range of approaches have been designed to reduce the repeated programming work and improve the reusability of other programs (e.g. the Component Object Model and plug-in architecture). Similarly, the development of geo-analysis models also follows these implementation approaches. In addition to the execution programs of geo-analysis models, some related external Dynamic Link Libraries (DLLs) are necessary to invoke execution; certain internally referenced model data (such as configure files and customized supportive databases) are also required.

Based on all of these characteristics of geo-analysis models, this paper designed a flexible and controllable model-deployment interface.

In Figure 3(a), the Unified Modelling Language (UML) diagram of the model-deployment interface (*IModelDeployment*) is introduced. There are four attributes in *IModelDeployment*: *Entrance* (in *ModelEntrance* type), *HardwareConfiguration* (a set of *ModelConfigures* that depict hardware-dependent information), *SoftwareConfiguration* (a set of *ModelConfigures* that depict software-dependent information), and *AssemblyConfiguration* (a set of *ModelAssemblys*).

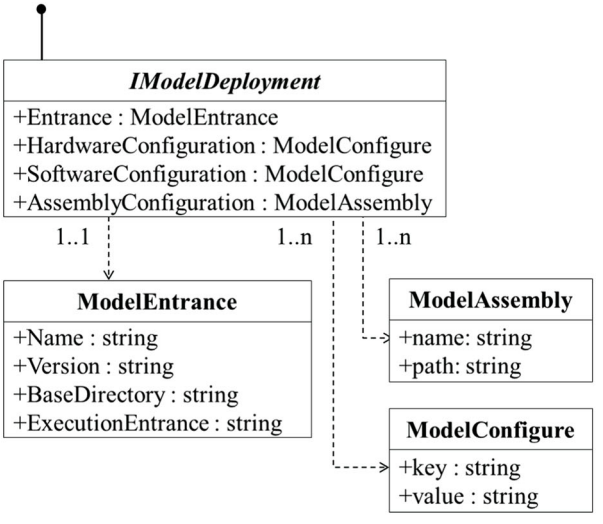
In *ModelEntrance*, *Name* indicates the execution program file's name; *Version* provides the version flag and its related web resource link; *BaseDirectory* gives the file system information for the execution file; and *ExecutionEntrance* describes the function name to invoke the model execution programs.

In *ModelConfigure*, *key* indicates the type name of the model-related resources (e.g. 'Platform' is the depended platform type and 'Runtime' is the depended programming language runtime) and *value* gives the name or file path information of the corresponding resource, including the software environment (e.g. database-software dependency, system runtime package), execution-related files (e.g. preload configure files, system demo files), model deployment help files, or website universal resource links (URLs).

In *ModelAssembly*, *path* means the directory of a model execution-related assembly file, and *name* describes the full name with the extension of this assembly file (.dll in WINDOWS environment, .so in LINUX, etc.).

In Figure 3(b), one of the TauDEM models (Terrain analysis using Digital Elevation Models, http://hydrology.usu.edu/taudem/taudem5/index.html) is employed to demonstrate the sample implementation of the deployment interface. A lightweight *Model Description Document* is employed to represent the corresponding information of a geo-analysis model. The designed model-deployment interface is serialized in Extensible Markup Language (XML) format. The root node is *ModelClass* (indicating the name and ID of a model); according to the design of the model-deployment interface, the deployment information is organized in the *Runtime* node. The *Runtime* node has three child nodes that represent hardware requirements (*HardwareConfigures* node), software requirements (*SoftwareConfigures* node), and model execution files (*Assemblies* node). Because different models require different hardware/software conditions, it is difficult to construct a comprehensive description system for the *key-value* set in *HardwareConfigures* and *SoftwareConfigures*. In the detailed implementation of *Model Description Document*, several predefined condition names for hardware (e.g. MemorySize, StorageSize, CPUCount, CPUInfomation, GPUInformation) and software (e.g. Platform, Runtime, 3rdPartySoftware, DataFile) are provided, and model providers can select specific items from them and then define the *key* parameter. Based on the *Model Description Document*, the information on required hardware conditions and dependent software environments can be organized in a structured manner and the execution-related files can be clearly described.

(a)



UML diagram of the model deployment interface

(b)

```
<ModelClass name="DInfFlowDirection"
    uid="{65CB9CDA-80C2-40B7-AF2A-74788BBD1576}">
  ......
  <Runtime name="DInfFlowDirection" version="5.0"
      baseDir="$installPath/Taudem5Exe/"
      entry="DinfFlowDir.exe">
    <HardwareConfigures>
      <Add key="MemorySize" value="1G" />
      <Add key="StorageSize" value="2.5M" />
    </HardwareConfigures>
    <SoftwareConfigures>
      <Add key="Platform" value="WINDOWS_32" />
      <Add key="Runtime" value="VC_2010" external="vcredist_x86.exe" />
    </SoftwareConfigures>
    <Assemblies>
      <Assembly name=".dll" path="mpich2.dll" />
      <Assembly name=".dll" path="mpich2mpe.dll" />
      <Assembly name=".dll" path="mpich2mpi.dll" />
      <Assembly name=".dll" path="mpich2mt.dll" />
      <Assembly name=".dll" path="mpich2mtp.dll" />
      <Assembly name=".dll" path="mpich2nemesis.dll" />
      <Assembly name=".dll" path="mpich2nemesisp.dll" />
      <Assembly name=".dll" path="mpich2p.dll" />
    </Assemblies>
  </Runtime>
</ModelClass>
```

Model Entrance

Hardware Configuration

Software Configuration

Assembly Configuration

Sample implementation of the model deployment interface

Figure 3. Design of the model-deployment interface.

3.1.2 Design of model-deployment packages

Because geo-analysis models are closely related to specific modelling backgrounds and disciplinary knowledge, the detailed implementations of geo-analysis models can vary and the input and output data formats can differ. These heterogeneities must be reduced to provide model users a more convenient way to reuse different geo-analysis models. This work can be referred to as model encapsulation and is aimed at mitigating original execution differences.

There have been numerous studies on combining original model execution programs into new 'components' (with the same calling and data input/output methods) (e.g. David et al. 2013; Wu et al. 2015; Yue et al. 2015a). One prominent example is the WPS standard; by using Geographic Markup Language as the data format and abstracting the execution of the geo-analysis model as 'Processing', a range of model services can be built (such as the Python WPS project (PyWPS) and the 52° North project) (Latu et al. 2013). Another typical example comes from the Open Modelling Interface (OpenMI) Association, which links different models by using a global standard for exchanging data (Castronova, Goodall, and Ercan 2013). Through encapsulation, they invoke a variety of geo-analysis models in a uniform manner.

This study attempts to develop a strategy based on encapsulation to deploy geo-analysis models as available model services. It has been widely accepted that using structured zip files to build installation packages can effectively redistribute software. Based on this idea, some researchers have studied moving-code packages that contain a description of the provided geo-processing logic and the platform requirements as well as their specific implementation (Müller, Bernard, and Brauner 2010); further research has extended this method to support the description of PaaS, IaaS, and licenses, along with the ability to use arbitrary standards for functional description (Müller, Bernard, and Kadner 2013). In borrowing this software engineering idea, this paper reorganizes the contents of the deployment package into two levels: (1) the model-core deployment level; and (2) the model-service-utility level. As seen in Figure 4, the model-core level primarily relates to model execution and the model-service-utility level primarily relates to model services.

The model-core level of the model-deployment package includes the model-encapsulation files and the *Model Description Document*. Figure 4 uses the same sample model in Figure 3 to illustrate the model-deployment package. The original *DinfFlowDir* execution program is wrapped to expose a set of uniformed invoking methods so that different models can be reused in the same manner (Yue et al. 2015b). At the top of Figure 4, the *$Encapsulation_DInfFlowDirection.dll* is the encapsulated dynamic link library that can be invoked by other programs (using dynamic load technology). The *$Encapsulation_DInfFlowDirection.mdl* is the *Model Description Document* explained in Section 3.1.1. In the model-core deployment level, the model-deployment interface is implemented to organize model execution-related resources in a structural manner.

In the model-service-utility level, except for the model execution-related files, other attached information is provided to help more completely understand the deployed model service. The *Supportive* fold primarily includes the installation files or URLs contained in the *SoftwareConfigures* node (as explained in the *Model Description Documentation*). Moreover, in the *Testify* fold, the demonstrate-data and a testify script are included to help verify whether a deployed model service can be used; this can also help model service users more clearly understand the input/output data of a model service and prepare appropriate data when using a specific model service. The testify-script file primarily records the demonstrated input file name and output file name of a model. In addition, a *license* document is included to explain the copyright of a model service.

By using the model-deployment interface, model providers can describe the information of the model deployment more clearly and structurally. Model users or model-service managers can prepare the model-deployment-related resources and 'install' a model service in a compute node more conveniently with the help of model-deployment packages. Thus, the collaborative barrier between model providers and model users can be reduced.

3.2 Computational resource description method

The execution-related environment of a geo-analysis model should be initially checked to see whether it can be deployed in a certain compute node. The hardware conditions and software environments of a compute node are described in the same manner as the design of the model-deployment interface. The model-deployment interface can be used to develop an automatic verification tool that compares the requirements of the ready-to-be-deployed model with the conditions

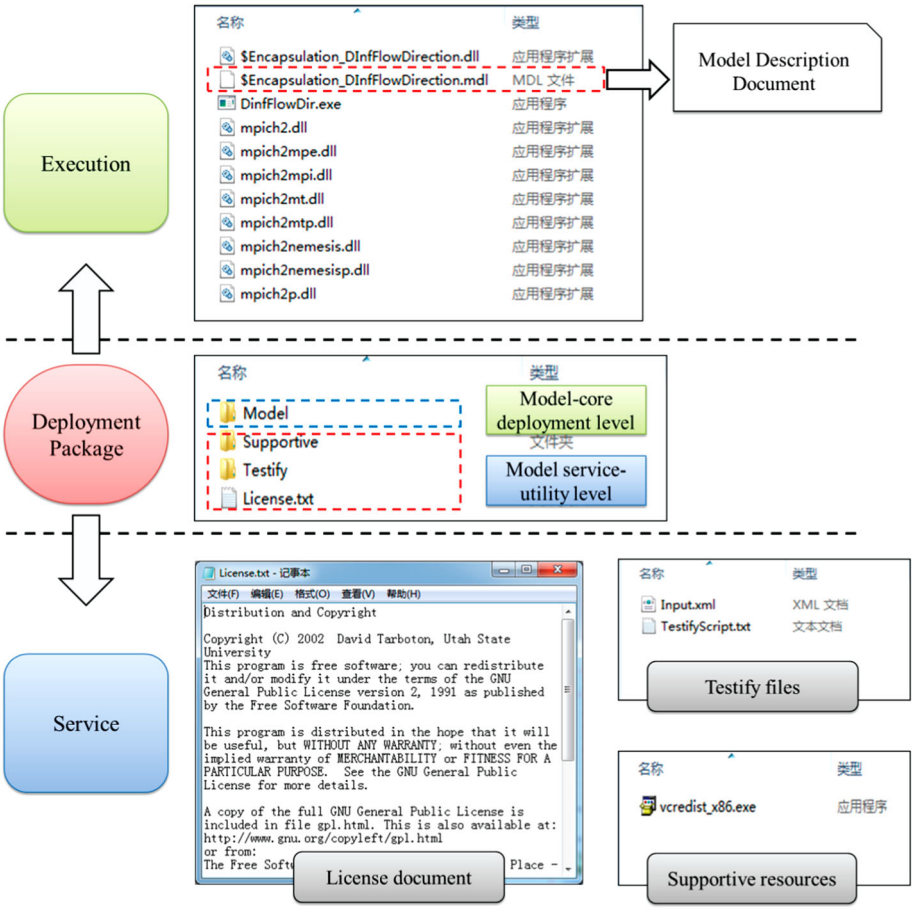


Figure 4. Design of the model-deployment package.

of a specific compute node. With regard to the hardware conditions, several Operating System APIs can be employed to enumerate the hardware parameters. Based on the *Model Description Document*, every hardware requirement can be exposed and compared with the corresponding hardware parameters. With regard to the software environments, the package list or system registry table for each software requirement can be visited to check whether specific software has been installed in the deployment target computer. If the deployment package contains the setup file for dependent software, it will be auto-installed; otherwise, the deployment request will return a failure notice back to the clients.

To deploy a geo-analysis model service in a compute node with an extremely flexible condition (provided by geographically dispersed individuals or groups), the model and computer should coordinate with each other. In addition to the construction of model-deployment packages, the method for making a normal computer an available server that can be used to deploy models and provide model services is also important.

With regard to the reusing of model services, the process can be outlined as shown in Figure 5. Two basic components are essential between the model users (clients) and the execution of a model (computation): the 'web server', which contains detailed execution, and the 'message transfer', which builds the communication bridge between clients and servers. Although the implementation methods of the geo-analysis model can vary (such as execution files, dynamic link libraries, or executable scripts), it is necessary to design a model-services container that creates and manages model

services from the deployment packages. In addition, the calling methods of various model services can also be different (e.g. SOAP, REST, WPS); this is primarily caused by the specific implementation of model encapsulation strategies. A model-services connector is needed to make it easy for model users to use these different model services, and the service-usage methods should be well described.

3.2.1 Model-service container

Based on the above analysis, an extendable service container interface is designed to manage the deployment of geo-analysis models and the published model services. The function structure of the model-service container is presented in Figure 6. A compute node should have the capability to communicate with model providers (so that model-deployment package can be uploaded and managed) and model-service users (so that model services can be called and then invoked). Thus, the information on the deployment server is exposed to make it possible for model providers to connect with compute nodes; the interface of managed deployment packages is offered for model providers to control their own resources (e.g. update old packages or upload new packages). Similarly, the information on the model-service server is exposed to provide the entrance for model users to use model services, and the interface of managed model services is offered to manage the model services published in a compute node (e.g. stop or delete a model service). In addition, the method of creating model services is implemented to analyse the model-deployment packages and then generate available services.

Based on the analysis of a service container's function structure, the basic design of a model-service container is presented in Figure 7. *IModelServiceManager* is the interface that should be implemented in a compute node to convert it from a normal computer to an available computational resource. *IModelServiceManager* includes the *deploymentServerList* (indicating the information on the deployment server), the *deploymentManager* (handling the manage logic of the deployment packages), the *modelServiceServerList* (indicating the information on the model-service server), and the *modelServiceManager* (handling the manage logic of the model-services). *createService()* indicates the method for unzipping a deployment package and publishing it as a service. The hardware and software conditions should be checked within the *createService()* process, and the model service can be created and published only if the check results satisfy the demands of the model-deployment requirements. Both the deployment server and model-service server are in list type so that various geo-analysis models can be organized by categories. The *deploymentServerList* and *modelServiceServerList* are described by *ServerInformation*, which primarily has the attributes *ipAddress* (a series of numbers describing the Internet Protocol address of a model service), *port* (describing the port number of a model service), and *extensive* (containing auxiliary information about the computational server). The *deploymentManager* is in *IDeploymentManager* type, which primarily includes the methods *Submit* (to add a model package), *Update* (to update an existing model package), *Query* (to obtain the information of a model package with some query conditions), and *Delete* (to remove a

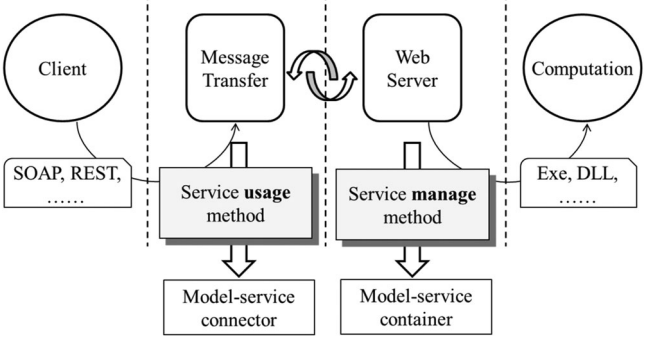


Figure 5. Basic process of using model services.

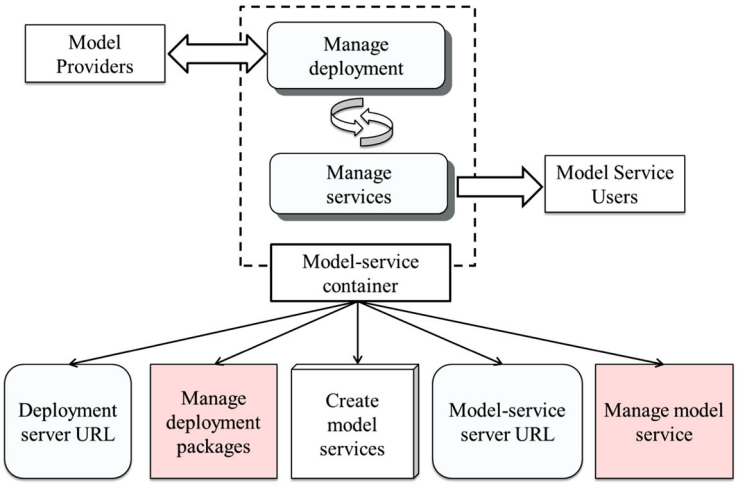


Figure 6. Analysis of the function structure of model-service container.

model package). The *modelServiceManager* is in *IServiceManager* type, which primarily includes the methods *Register* (to publish a new model service), *Update* (to update current existing model services), *Query* (to obtain the information of a model service with some query conditions), *Delete* (to remove an existing model service), *Start* (to make a model service available to model users across the internet), *Stop* (to make a model service unavailable to model users), and *Pause* (to make a model service currently unavailable).

Service deployment work can be more flexibly controlled with the model-service container, and the deployment packages can be more conveniently converted to available model services.

3.2.2 Model-service connector

Because different geo-analysis models are implemented by various technological approaches, the usage of a model service can be diverse (based on the WPS standard, using SOAP or RESTful

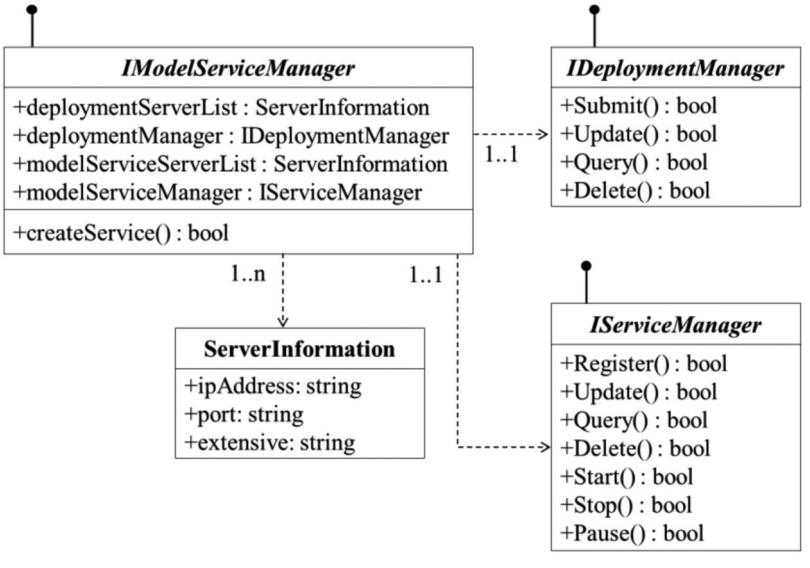


Figure 7. Basic design of the model-service container.

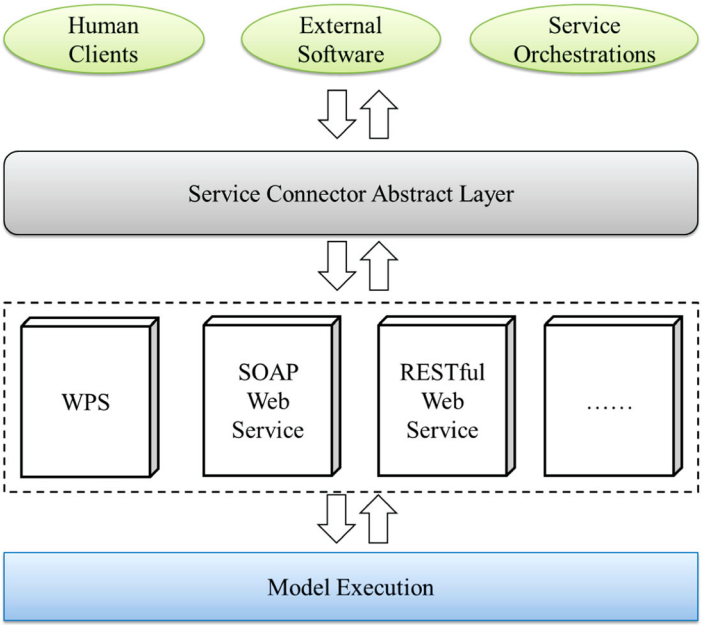


Figure 8. Basic application scene of model services.

style). Figure 8 shows the basic model-service application scene. The model services can also be used by different types of user clients that depend on specific modelling targets such as use by human individuals, integrating with other software, and composing according to certain service orchestration scripts. Therefore, this study designed a Service Connector Abstract Layer between various model-service manners and different service-user clients.

This abstract layer primarily processes the diverse model-service request/response operations. *IModelServiceConnector* is designed to describe the operating instructions of a model service (see Figure 9). In the *IModelServiceConnector* interface, the *wkName* is designed to provide the well-known name of the operation method for a model service (e.g. the *wkName* of WPS indicates that the operation method should follow to the WPS standard: *GetCapabilities*, *DescribeProcess*, *Execution*, or *GetStatus*). The *wiki* attribute would give the URL for a specific execution style (e.g. the website of the WPS standard would be provided by the wiki attribute if the operation method of a model service follows the WPS instruction). The *operationList* attribute is designed to provide the request methods of a model service and the corresponding respond content (such as the *DescribeProcess* operation and the process description document in the WPS standard). *operationList* is in *OperationInformation* type, which primarily describes the *Name* (indicating the request name), *Verb* (e.g. POST and GET), *Request* (indicating the request URL), and *Response* (indicating the response document).

The *executionStyle* attribute in the *IModelServiceConnector* interface describes the invocation method of the model services. Because geo-analysis models are developed to depict complex environmental processes and the execution of geo-analysis models typically involves a range of intermediate steps, this study employed the concept 'state' to describe the execution process. Using a series of 'states', a model service can be clearly presented to model users and the detailed execution status can be returned to model users. The execution manner of a model service is concluded as an enumeration: *EExecutionStyle*. There are three execution styles: *StateSimulation* style, *SimpleCalculation* style, and *TimeSeries* style. Figure 9(b) presents the relation of these three styles, and the concept 'state' is demonstrated as *node*. With regard to the *StateSimulation* style, the model service runs as a workflow that consists of a series of *nodes*. There is 1 *start-node* (indicating starting to use a model service) and 1 *end-node* (indicating the execution has finished) within the entire execution

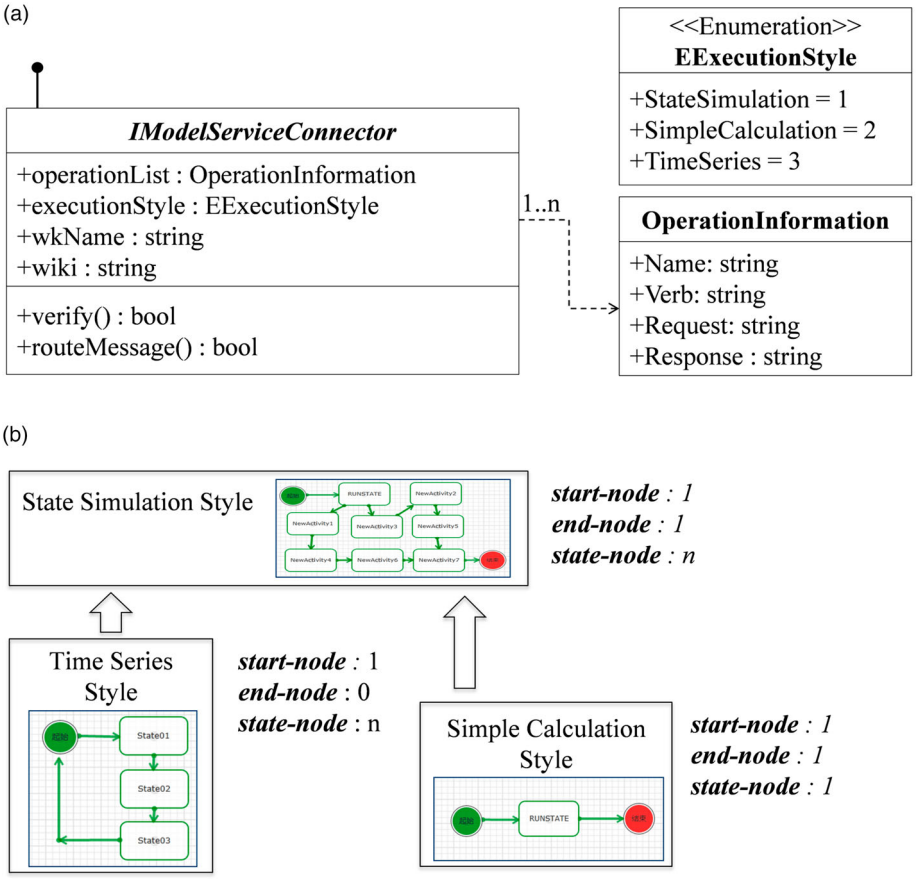


Figure 9. Basic design of the model-service connector interface.

process and a range of *state-nodes* to perform detailed computation. With regard to the *SimpleCalculation* style, it is a special *StateSimulation* with only 1 *start-node*, 1 *end-node*, and 1 *state-node*. The *SimpleCalculation* style model service is widely used in geo-processing; typical examples include WPS-based model services such as computing buffer areas of feature datasets and computing a region's slope from DEM. With regard to the *TimeSeries* style, it is also a special *StateSimulation* that has only 1 *start-node*, a range of *state-nodes*, and no *end-node*. The execution process of model services in this style will restart once all of the *state-nodes* have been executed.

In addition, there are two methods in the *IModelServiceConnector*: *verify()* and *routeMessage()*. The method *verify()* is implemented to check if the requested URL string is correct, which means that it should match with a specified item in the *operationList*. The method *routeMessage()* is implemented to transfer the request of the model services and the posted data into the actual execution of geo-analysis models.

With the proposed model-service connector, the deployment results of geo-analysis models can be published as available model services, and model users can more clearly understand how to use a model service according to the description information exposed by the model-service connector interface.

4. Experiments

A prototype open-modelling platform was developed to verify the capability and practicality of the proposed geo-analysis model-deployment method. Figure 10 introduces the basic architecture of this

modelling platform. The deployment process was conducted more easily based on the model-deployment description interface (implemented by model providers to describe model-deployment packages) and the computational resource description interface (implemented by computational resource providers to convert a compute node to an available model-service server and enable connection with model-service users).

A collaborative strategy was realized for providing geo-analysis models and computational resources while creating and reusing model services (see Figure 10). A portal website is employed to help modellers work together through a universal entrance. There are three repositories within this platform: the Deployment Package Repository, Model Resource Repository, and Computational Resource Repository.

The Deployment Package Repository is used to save and organize various model-deployment packages. The deployment process is presented in the left part of Figure 10. Through the portal website, model providers can (1) submit model-deployment packages into the Deployment Package Repository; (2) select a compute node from the Computation Resource Repository to deploy the model service; (3) transmit the deployment package to the selected compute node and create the corresponding model service; and (4) if the geo-analysis model can be deployed in the selected compute node and the model service has been successfully published, then the service can be registered into the Model Service Pool as an available model service.

The Computational Resource Repository is designed to record all of the available compute nodes within the platform. The computational resource providers should first implement the model-service container and model-service connector in their ready-to-be-offered compute nodes. Then, they can register the compute nodes as an available model-service server into the Computational Resource Repository (see the right part of Figure 10).

The Model Resource Repository is designed to store and organize various model resources. After a model-deployment package has been submitted to the Deployment Package Repository, an item that records this model needs to be created and stored in the Model Resource Repository. The model-service item in the Model Service Pool is then referenced by the unique identifier of a model existing in the

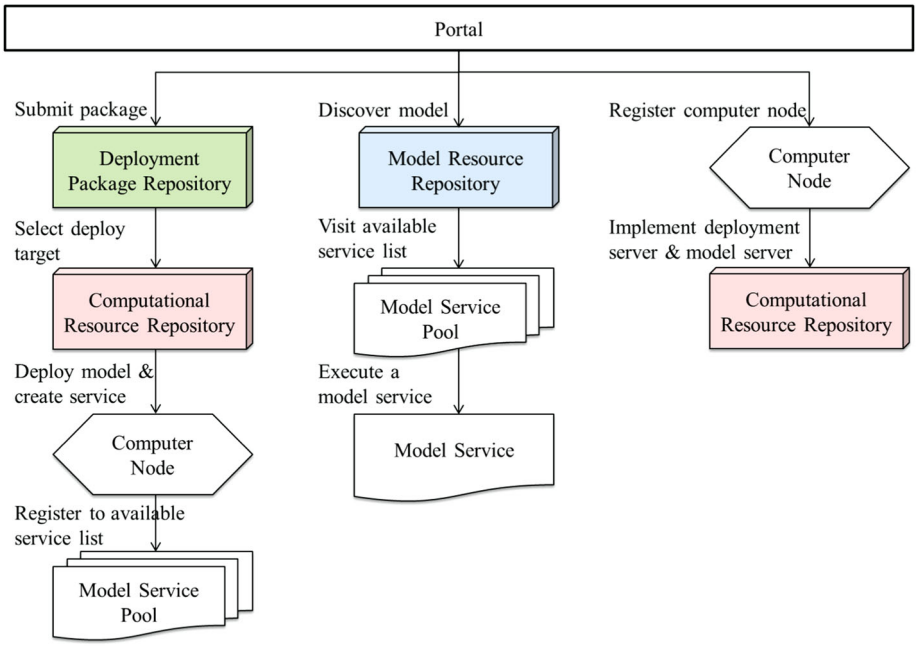


Figure 10. Basic design of the collaborative modelling platform.

Model Resource Repository. Because a model service can be published in multiple compute nodes, a model can reference several model services. Model users can (1) discover their relevant model in the Model Resource Repository; (2) visit all of the available model services that relate to the selected geo-analysis model; and (3) use a specific model service (see the middle part of Figure 10).

An experimental package tool was designed and developed to help model providers more conveniently build the model-deployment packages. Figure 11 shows that the experimental package tool primarily implements the model-deployment description interface; it is currently developed based on the .Net platform. As shown in Figure 11, the Soil and Water Assessment Tool (SWAT, http://swat.tamu.edu) model is employed to illustrate the package building process. To make it easier for model users to use the SWAT model, the original execution program was encapsulated based on the MapWindow SWAT (MWSWAT). Within the package tool, the encapsulated SWAT model was imported to the *Model Entrance*. The hardware and software conditions were inputted into the tool; the dependent assemblies were then auto-checked (some missing assemblies should still be imported by users). Thus, the corresponding *Model Description Document* was generated as shown in Figure 11(a). With regard to the service-utility level information, the presented package tool provides functions to import supportive resources, testify data, and license documents into the deployment package. Moreover, a testify-platform was also provided within the package tool that can check the

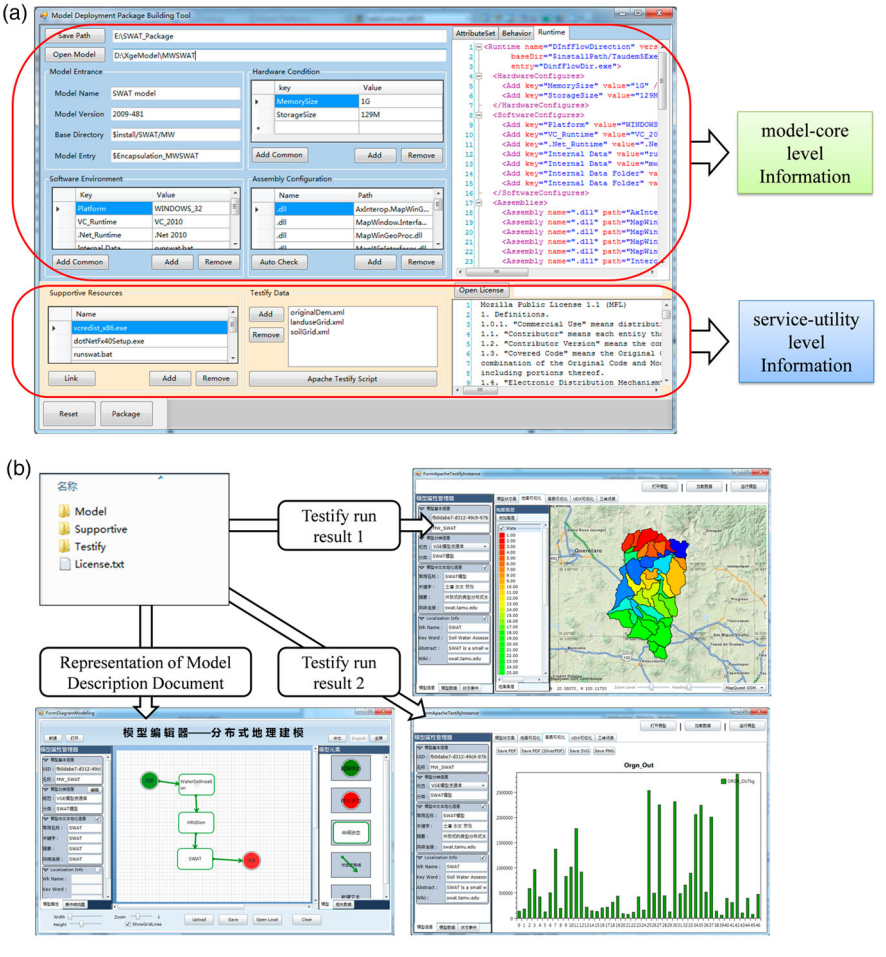


Figure 11. Prototype implementation of model-deployment package tool.

execution result of the ready-to-deployed model in real time. The results of the sample SWAT model are presented in Figure 11(b).

To address the compute node, this study also designed and developed an experimental model-service container execution program and an experimental model-service connector execution program. In Windows, this implementation is based on ASP.Net architecture; in Linux, the implementation is based on the Node.JS project. Figure 12 shows the basic process of deploying a model service in a compute node (Figure 12(a)), discovering a model resource in the model resource repository (Figure 12(b)), selecting a model service from the model-services pool (Figure 12(c)), and invoking an available model service (Figure 12(d)).

Based on the above analysis, this study builds a prototype system for sharing, reusing and executing geo-analysis models in an open web environment. Figure 13 presents some snapshots of this system (http://geomodeling.njnu.edu.cn). A simple and typical example was employed to explain the collaborative process when performing modelling by using resources provided by different stakeholders. As indicated in the top-left part of Figure 13, a 'China Sea Tide Simulation' model was provided by a model resource provider. By using the package tool introduced in Figure 11, the deployment package for this model was built. The model-resource provider described the information related to the model through the portal website and then uploaded the deployment package into the Deployment Package Repository. As shown in the top-right part of Figure 13, a computational resource provider installed the model-service container and connector in the ready-to-be-shared computer node (using the Internet Information Service tool on the Windows platform), described the basic information concerning the computer node, and registered it into the Computational Resource Repository. Through the portal website, the 'China Sea Tide Simulation' model could then be downloaded to the computer node as a deployment package to constitute the available model service. As shown at the bottom of Figure 13, a model user discovered this model service from the list of model services on the port website. The model user prepared the input data for the model and built a modelling case by invoking the execution of this model service. Therefore, the model

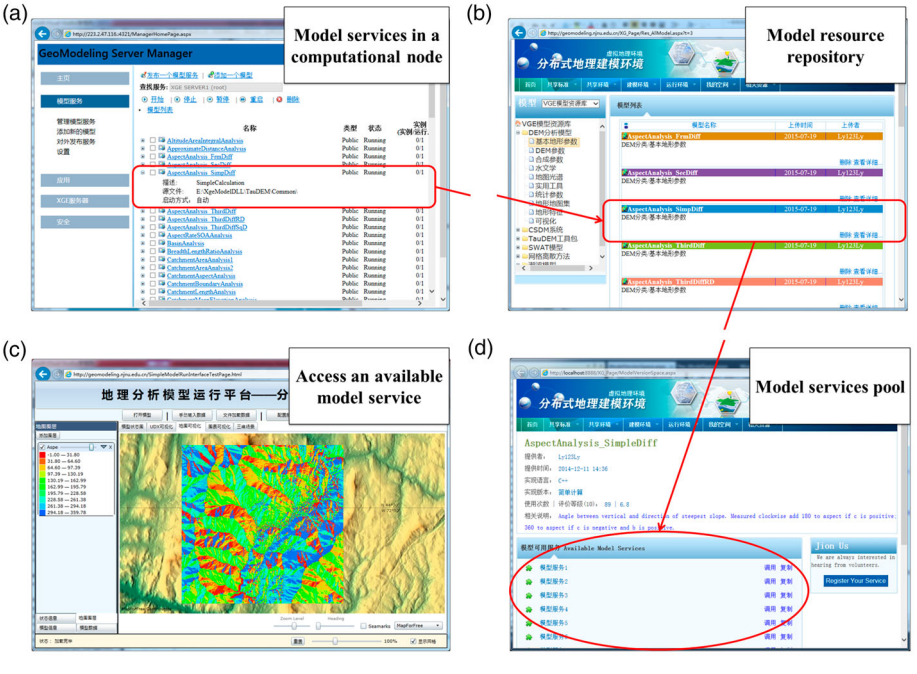


Figure 12. Experimental implementation of model-service management tool.

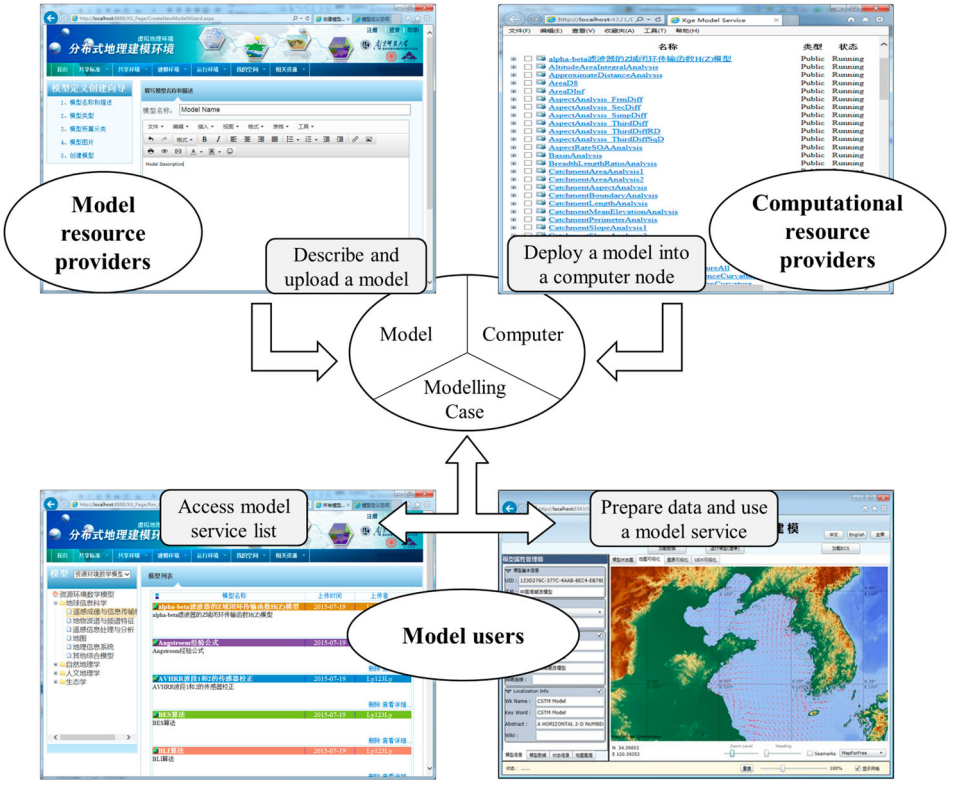


Figure 13. Prototype system for sharing and integrating geo-analysis models in open web environment.

resource provider, the computational resource provider, and the model user collaboratively worked together to build the modelling case.

5. Conclusions and future work

This study proposed reducing the difficulty in deploying model services in an open web environment, which is important in the reusing, sharing, and integrating of geo-analysis models. Using the proposed model-deployment strategy, model resource providers and computational resource providers can more conveniently and collaboratively work together. With the proposed model-deployment description interface, the hardware, software, and model-related information can be described in a flexible and extendable manner, and model-deployment packages can be built. Based on the computational resource description interface, model-service containers and model-service connectors can be implemented, thus making it more convenient for modellers to deploy various geo-analysis models in an appropriate compute node. Based on the proposed model-deployment method, modellers from dispersed regions can more easily work together and provide more dynamic and reliable geospatial information for the study of Future Earth. Moreover, the proposed method can provide a promising, effective means for researchers to work collaboratively in building a Digital Earth platform to simulate a complex and synthetic earth environment (e.g. global changes, geographical processes with human impacts).

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

(1) More complete and automated tools are needed to deploy model services. The technology and architecture of geo-analysis models vary between different model developers. The proposed

model-deployment strategy can provide a clear and structured method to describe the deployment-related information of different geo-analysis models, but the selection of the appropriate compute nodes to deploy models still requires manual effort. By interpreting the deployment information, a more automated method of selecting compute nodes is needed to satisfy the demands of building an extendable model library for dynamic simulations of earth environments in the study of Future Earth.

(2) Different web transmission methods should be considered to use model services and communicate with distributed model servers. After deploying geo-analysis models, model services can be created and published. To use these model services more conveniently and effectively, the data transmission and message communication between user clients and model-service servers should be coordinated. In addition to the general HTTP method, other methods (e.g. P2P and WebSocket) should be employed to develop more useful modelling platforms.

(3) Based on the proposed Model Resource Repository, Deployment Package Repository, and Computational Resource Repository, other applications (such as the service-composition, service-orchestration, and geo-processing webs) can be built. QoS technology can be applied to support more intelligent model-service selection. The Geo-Processing Workflow (GPW), Business Processing Execution Language (BPEL), and other service-composition methods can be used to build scientific workflows; the web crawler method can be developed to help discover interested model resources.

Acknowledgements

We appreciate the detailed suggestions and comments from the editor and the anonymous reviewers.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Basic Research Program of China (973 Program) under Grant number 2015CB954102; the National Natural Science Foundation of China under Grant number 41471317, Grant number 41301414 and Grant number 41371424; and the Priority Academic Program Development of the Jiangsu Higher Education Institutions under Grant number 164320H116.

References

Arciniegas, G., R. Janssen, and P. Rietveld. 2013. "Effectiveness of Collaborative Map-Based Decision Support Tools: Results of an Experiment." *Environmental Modelling & Software* 39: 159–175. doi:10.1016/j.envsoft.2012.02.021.

Baranski, B., T. Foerster, B. Schäffer, and K. Lange. 2011. "Matching INSPIRE Quality of Service Requirements with Hybrid Clouds." *Transactions in GIS* 15 (s1): 125–142. doi:10.1111/j.1467-9671.2011.01265.x.

Basnyat, P., L. D. Teeter, B. G. Lockaby, and K. M. Flynn. 2000. "The Use of Remote Sensing and GIS in Watershed Level Analyses of Non-Point Source Pollution Problems." *Forest Ecology and Management* 128 (1): 65–73. doi:10.1016/S0378-1127(99)00273-X.

Brovelli, M. A., S. Li, S. Dragicevic, and B. Veenendaal. 2013. "Introductory Editorial: Web-Based Sensors and Geoprocessing Services." *Applied Geomatics* 5 (1): 1–2. doi:10.1007/s12518-013-0102-z.

Butt, M. A., and S. Li. 2012. "Developing a Web-Based, Collaborative PPGIS Prototype to Support Public Participation." *Applied Geomatics* 4 (3): 197–215. doi:10.1007/s12518-012-0085-1.

Castronova, A. M., J. L. Goodall, and M. B. Ercan. 2013. "Integrated Modeling Within a Hydrologic Information System: An Open MI Based Approach." *Environmental Modelling & Software* 39: 263–273. doi:10.1016/j.envsoft.2012.02.011.

Chang, Z. E., and S. Li. 2013. "Geo-Social Model: A Conceptual Framework for Real-Time Geocollaboration." *Transactions in GIS* 17 (2): 182–205. doi:10.1111/j.1467-9671.2012.01352.x.

Chen, J., Y. Ban, and S. Li. 2014. "China: Open Access to Earth Land-Cover map." *Nature* 514 (7523): 434–434. doi:10.1038/514434c.

Chen, A., L. Di, Y. Wei, Y. Bai, and Y. Liu. 2009. "Use of Grid Computing for Modeling Virtual Geospatial Products." *International Journal of Geographical Information Science* 23 (5): 581–604. doi:10.1080/13658810902733666.

Chen, M., H. Lin, M. Hu, H. Li, and C. Zhang. 2013a. "Real Geographic Scenario Based Virtual Social Environment: Integrate Geography with Social Research." *Environment and Planning B-Planning & Design* 40 (6): 1103–1121. doi:10.1068/b38160

Chen, M., H. Lin, O. Kolditz, and C. Chen. 2015. "Developing Dynamic Virtual Geographic Environments (VGEs) for Geographic Research." *Environmental Earth Sciences* 74: 6975–6980. doi:10.1007/s12665-015-4761-4.

Chen, J., H. Wu, S. Li, A. Liao, C. He, and S. Peng. 2013b. "Temporal Logic and Operation Relations Based Knowledge Representation for Land Cover Change Web Services." *ISPRS Journal of Photogrammetry and Remote Sensing* 83: 140–150. doi:10.1016/j.isprsjprs.2013.02.005.

Claussen, M., L. Mysak, A. Weaver, M. Crucifix, T. Fichefet, M. F. Loutre, L. Weber, et al. 2002. "Earth System Models of Intermediate Complexity: Closing the Gap in the Spectrum of Climate System Models." *Climate Dynamics* 18 (7): 579–586. doi:10.1007/s00382-001-0200-1.

Craglia, M., M. F. Goodchild, A. Annoni, G. Camara, M. Gould, W. Kuhn, D. Mark, 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. doi: 10.2902/1725-0463.2008.03.art9.

David, O., J. C. Ascough, W. Lloyd, T. R. Green, K. W. Rojas, G. H. Leavesley, and L. R. Ahuja. 2013. "A Software Engineering Perspective on Environmental Modeling Framework Design: The Object Modeling System." *Environmental Modelling & Software* 39: 201–213. doi:10.1016/j.envsoft.2012.03.006.

Di Giovanni, P., M. Bertolotto, G. Vitiello, and M. Sebillo. 2014. "Web Services Composition and Geographic Information." In *Geographical Information Systems: Trends and Technologies,* edited by E. Pourabbas, 104–141. Boca Raton: Taylor and Francis Group.

Dickinson, R. E., K. W. Oleson, G. Bonan, F. Hoffman, P. Thornton, M. Vertenstein, Z. L. Yang, and X. B. Zeng. 2006. "The Community Land Model and its Climate Statistics as a Component of the Community Climate System Model." *Journal of Climate* 19 (11): 2302–2324. doi:10.1175/JCLI3742.1.

Dobbs, C., D. Kendal, and C. R. Nitschke. 2014. "Multiple Ecosystem Services and Disservices of the Urban Forest Establishing Their Connections with Landscape Structure and Sociodemographics." *Ecological Indicators* 43: 44–55. doi:10.1016/j.ecolind.2014.02.007.

Evangelidis, K., K. Ntouros, S. Makridis, and C. Papatheodorou. 2014. "Geospatial Services in the Cloud." *Computers & Geosciences* 63: 116–122. doi:10.1016/j.cageo.2013.10.007.

Foster, I., Y. Zhao, I. Raicu, and S. Lu. 2008. "Cloud Computing and Grid Computing 360-Degree Compared." Grid Computing Environments Workshop, 2008. GCE '08, Austin, November 12–16. doi:10.1109/GCE.2008.4738445.

Giuliani, G., S. Nativi, A. Lehmann, and N. Ray. 2012. "WPS Mediation: An Approach to Process Geospatial Data on Different Computing Backends." *Computers & Geosciences* 47: 20–33. doi:10.1016/j.cageo.2011.10.009.

Goodchild, M. F. 2008. "The Use Cases of Digital Earth." *International Journal of Digital Earth* 1 (1): 31–42. doi:10.1080/17538940701782528.

Goodchild, M. F. 2012. "The Future of Digital Earth." *Annals of GIS* 18 (2): 93–98. doi:10.1080/19475683.2012.668561.

Goodchild, M. F., H. Guo, A. Annoni, L. Bian, K. Bie, F. Campbel, M. Cragliac, et al. 2012. "Next-Generation Digital Earth." *Proceedings of the National Academy of Sciences* 109 (28): 11088–11094. doi:10.1073/pnas.1202383109.

Granell, C., L. Díaz, and M. Gould. 2010. "Service-Oriented Applications for Environmental Models: Reusable Geospatial Services." *Environmental Modelling & Software* 25 (2): 182–198. doi:10.1016/j.envsoft.2009.08.005.

Granell, C., S. Schade, and N. Ostländer. 2013. "Seeing the Forest Through the Trees: A Review of Integrated Environmental Modelling Tools." *Computers, Environment and Urban Systems* 41: 136–150. doi:10.1016/j.compenvurbsys.2013.06.001.

Griggs, D., M. Stafford-Smith, O. Gaffney, J. Rockström, M. C. Öhman, P. Shyamsundar, W. Steffen, et al. 2013. "Policy: Sustainable Development Goals for People and Planet." *Nature* 495 (7441): 305–307. doi:10.1038/495305a.

Guo, H., Z. Liu, and L. Zhu. 2010. "Digital Earth: Decadal Experiences and Some Thoughts." *International Journal of Digital Earth* 3 (1): 31–46. doi:10.1080/17538941003622602.

Gutierrez, F., M. Parise, J. D. Waele, and H. Jourde. 2013. "A Review on Natural and Human-Induced Geohazards and Impacts in Karst." *Earth Science Review* 138: 61–88. doi:10.1016/j.earscirev.2014.08.002.

Han, G., J. Chen, C. He, S. Li, H. Wu, A. Liao, and S. Peng. 2015. "A Web-Based System for Supporting Global Land Cover Data Production." *ISPRS Journal of Photogrammetry and Remote Sensing* 103: 66–80. doi:10.1016/j.isprsjprs.2014.07.012.

Hofer, B. 2014. "Uses of Online Geoprocessing Technology in Analyses and Case Studies: A Systematic Analysis of Literature." *International Journal of Digital Earth* 8: 901–917. doi:10.1080/17538947.2014.962632.

Hou, D., J. Chen, H. Wu, S. Li, F. Chen, and W. Zhang. 2015. "Active Collection of Land Cover Sample Data from Geo-Tagged Web Texts." *Remote Sensing* 7 (5): 5805–5827. doi:10.3390/rs70505805.

Jørgensen, S. E. 2015. "Structurally Dynamic Models: A New Promising Model Type." *Environmental Earth Sciences,* 74 (10): 7041–7048. doi:10.1007/s12665-015-4735-6.
Khan, Z., D. Ludlow, R. McClatchey, and A. Anjum. 2012. "An Architecture for Integrated Intelligence in Urban Management Using Cloud Computing." *Journal of Cloud Computing* 1 (1): 1–14. doi:10.1186/2192-113X-1-1.
Krämer, M., and I. Senner. 2015. "A Modular Software Architecture for Processing of Big Geospatial Data in the Cloud." *Computers & Graphics* 49: 69–81. doi:10.1016/j.cag.2015.02.005.
Laniak, G. F., G. Olchin, J. Goodall, A. Voinov, M. Hill, P. Glynn, G. Whelan, et al. 2013. "Integrated Environmental Modeling: A Vision and Roadmap for the Future." *Environmental Modelling & Software* 39: 3–23. doi:10.1016/j.envsoft.2012.09.006.
Lartigau, J., X. Xu, L. Nie, and D. Zhan. 2015. "Cloud Manufacturing Service Composition Based on QoS with Geo-Perspective Transportation Using an Improved Artificial Bee Colony Optimisation Algorithm." *International Journal of Production Research* 53: 4380–4404. doi:10.1080/00207543.2015.1005765.
Latu, K., N. Swain, S. Christensen, N. Jones, E. Nelson, and G. Williams. 2013. "Essential GIS Technologies for Hydrologic Simulation Applications in Cloud Computing." *World Environmental and Water Resources Congress* 2013: 2758–2767. doi:10.1061/9780784412947.273.
Li, Y., J. Gong, Y. Song, Z. Liu, T. Ma, H. Liu, S. Shen, W. Li, and Y. Yu. 2015. "Design and Key Techniques of a Collaborative Virtual Flood Experiment that Integrates Cellular Automata and Dynamic Observations." *Environmental Earth Sciences* 74 (10): 7059–7067. doi:10.1007/s12665-015-4716-9.
Li, S., C. Xiong, and Z. Ou. 2011. "A Web GIS for Sea Ice Information and an Ice Service Archive." *Transactions in GIS* 15 (2): 189–211. doi:10.1111/j.1467-9671.2011.01250.x.
Li, H., Q. Zhu, X. Yang, and L. Xu. 2012. "Geo-Information Processing Service Composition for Concurrent Tasks: A QoS-Aware Game Theory Approach." *Computers & Geosciences* 47: 46–59. doi:10.1016/j.cageo.2011.10.007.
Lin, H., M. Batty, S. E. Jørgensen, B. Fu, M. Konecny, A. Voinov, P. Torrens, et al. 2015. "Virtual Environments Begin to Embrace Process-Based Geographic Analysis." *Transactions in GIS* 19 (4): 493–498. doi:10.1111/tgis.12167.
Lin, H., M. Chen, and G. Lu. 2013a. "Virtual Geographic Environment: A Workspace for Computer-Aided Geographic Experiments." *Annals of the Association of American Geographers* 103 (3): 465–482. doi:10.1080/00045608.2012.689234.
Lin, H., M. Chen, G. Lu, Q. Zhu, J. Gong, X. You, Y. Wen, B. Xu, and M. Hu. 2013b. "Virtual Geographic Environments (VGEs): A New Generation of Geographic Analysis Tool." *Earth-Science Reviews* 126: 74–84. doi:10.1016/j.earscirev.2013.08.001.
Luo, W., and A. M. MacEachren. 2014. "Geo-Social Visual Analytics." *Journal of Spatial Information Science* 8: 27–66. doi:10.5311/JOSIS.2014.8.139.
Luo, W., P. Yin, Q. Di, F. Hardisty, and A. M. MacEachren. 2014. "A Geovisual Analytic Approach to Understanding Spatial–Social Relationships in the International Trade Network." *PLoS ONE* 9 (2) e88666. doi:10.1371/journal.pone.0088666.
Matheus, A. 2013. "Security Considerations on Processing of Geospatial Information in the Cloud." Computing for Geospatial Research and Application (COM. Geo), San Jose, July 22–24. doi:10.1109/COMGEO.2013.13.
Miller, H. J., and M. F. Goodchild. 2014. "Data-driven Geography." *Geo Journal* 80 (4): 449–461. doi:10.1007/s10708-014-9602-6.
Müller, M. 2015. "Hierarchical Profiling of Geoprocessing Services." *Computers & Geosciences* 82: 68–77. doi:10.1016/j.cageo.2015.05.017.
Müller, M., L. Bernard, and J. Brauner. 2010. "Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms." *Transactions in GIS* 14 (s1): 101–118. doi:10.1111/j.1467-9671.2010.01205.x.
Müller, M., L. Bernard, and D. Kadner. 2013. "Moving Code – Sharing Geoprocessing Logic on the Web." *ISPRS Journal of Photogrammetry and Remote Sensing* 83: 193–203. doi:10.1016/j.isprsjprs.2013.02.011.
Parsons, M. A. 2011. "Making Data Useful for Modelers to Understand Complex Earth Systems." *Earth Science Informatics* 4 (4): 197–223. doi:10.1007/s12145-011-0089-0.
Reichardt, M. 2010. "Open Standards-Based Geoprocessing Web Services Support the Study and Management of Hazard and Risk." *Geomatics, Natural Hazards and Risk* 1 (2): 171–184. doi:10.1080/19475701003643458.
Schaeffer, B. 2008. "Towards a Transactional WPS (WPS-T)." In *GI-Days 2008: Proceedings of the Sixth Geographic Information Days,* edited by E. Pebesma, M. Bishr and T. Bartoschek, 91–116. Münster: Institut für Geoinformatik.
Shi, Y., A. Shortridge, and J. Bartholic. 2002. "Grid Computing For Real Time Distributed Collaborative Geoprocessing." In *Geospatial Theory, Processing and Applications (ISPRS Technical Commission IV Symposium)*, ISPRS, 197–208, July 9–12.
Skidmore, A., and P. Woodgate. 2012. "Digital Earth 2020: Towards the Vision for the Next Decade." *International Journal of Digital Earth* 5 (1): 4–21. doi:10.1080/17538947.2011.638500.
Steiniger, S., and A. J. Hunter. 2012. "Free and Open Source GIS Software for Building a Spatial Data Infrastructure." In *Geospatial Free and Open Source Software in the Twenty-First Century,* edited by E. Bocher and M. Neteler, 247–261. Heidelberg: Springer, LNG&C.

Thakker, S., J. Abdul, and M. B. Potdar. 2015. "GeoProcessing Workflow Models for Distributed Processing Frameworks." *International Journal of Computer Applications* 113 (1): 33–38. doi:10.5120/19794-1574

Todorova, A., D. Syrakov, G. Gadjhev, G. Georgiev, K. G. Ganev, M. Prodanova, N. Miloshev, V. Spiridonov, A. Bogatchev, and K. Slavov. 2010. "Grid Computing for Atmospheric Composition Studies in Bulgaria." *Earth Science Informatics* 3 (4): 259–282. doi:10.1007/s12145-010-0072-1.

Torrens, P. M. 2015. "Intertwining Agents and Environments." *Environmental Earth Sciences* 74 (10): 7117–7131. doi:10.1007/s12665-015-4738-3.

Torrens, P. M., and A. McDaniel. 2013. "Modeling Geographic Behavior in Riotous Crowds." *Annals of the Association of American Geographers* 103 (1): 20–46. doi:10.1080/00045608.2012.685047.

Tsai, W. T., X. Sun, and J. Balasooriya. 2010. "Service-Oriented Cloud Computing Architecture." In *Information Technology: New Generations (ITNG)*, 2010 Seventh International Conference, Las Vegas, April 12–14.

Veenendaal, B., S. Li, S. Dragicevic, and M. A. Brovelli. 2014. "Analytical Geospatial Digital Earth." *International Journal of Digital Earth,* 7 (4): 253–255. doi:10.1080/17538947.2014.896967.

Wang, X., Z. Cheng, Z. Zhou, K. Ning, and L. Zhang. 2014. "Geospatial Web Service Sub-Chain Ranking and Recommendation." In *Services Computing (SCC)*, 2014 IEEE International Conference, Anchorage, June 27–July 2.

Wen, Y., M. Chen, G. Lu, H. Lin, L. He, and S. Yue. 2013. "Prototyping an Open Environment for Sharing Geographical Analysis Models on Cloud Computing Platform." *International Journal of Digital Earth* 6 (4): 356–382. doi:10.1080/17538947.2012.716861.

Wu, H., L. You, Z. Gui, S. Gao, Z. Li, and J. Yu. 2014. "FAST: A Fully Asynchronous and Status-Tracking Pattern for Geoprocessing Services Orchestration." *Computers & Geosciences* 70: 213–228. doi:10.1016/j.cageo.2014.06.005.

Wu, H., L. You, Z. Gui, K. Hu, and P. Shen. 2015. "GeoSquare: Collaborative Geoprocessing Models' Building, Execution and Sharing on Azure Cloud." *Annals of GIS* 21 (4): 287–300. doi:10.1080/19475683.2015.1098727.

Yang, C., M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacusf, and D. Fay. 2011. "Spatial Cloud Computing: How Can the Geospatial Sciences Use and Help Shape Cloud Computing?" *International Journal of Digital Earth* 4 (4): 305–329. doi:10.1080/17538947.2011.587547.

Yang, C., Z. Xie, and Z. Xu. 2014. "An Asynchronous Geoprocessing Workflow and Its Application to an Antarctic Ozone Monitoring and Mapping Service." *International Journal of Digital Earth.* Advance online publication: 1–15. doi:10.1080/17538947.2014.967318.

Yang, C., Y. Xu, and D. Nebert. 2013. "Redefining the Possibility of Digital Earth and Geosciences with Spatial Cloud Computing." *International Journal of Digital Earth,* 6 (4): 297–312. doi:10.1080/17538947.2013.769783.

Yue, S., M. Chen, Y. Wen, and G. Lu. 2015a. "Service-Oriented Model-Encapsulation Strategy for Sharing and Integrating Heterogeneous Geo-Analysis Models in an Open Web Environment." *ISPRS Journal of Photogrammetry and Remote Sensing.* doi:10.1016/j.isprsjprs.2015.11.002.

Yue, S., Y. Wen, M. Chen, G. Lu, D. Hu, and F. Zhang. 2015b. "A Data Description Model for Reusing, Sharing and Integrating Geo-Analysis Models." *Environmental Earth Sciences* 74 (10): 7081–7099. doi:10.1007/s12665-015-4270-5.

Yue, P., H. Zhou, J. Gong, and L. Hu. 2013. "Geoprocessing in Cloud Computing Platforms – A Comparative Analysis." *International Journal of Digital Earth,* 6 (4): 404–425. doi:10.1080/17538947.2012.748847.

Zhang, C., M. Chen, R. R. Li, C. Y. Fang, and H. Lin. 2015. "What's Going on About Geo-Process Modeling in Virtual Geographic Environments (VGEs)." *Ecological Modelling* 319: 147–154. doi:10.1016/j.ecolmodel.2015.04.023.

Zhang, C., T. Zhao, and W. Li. 2010. "The Framework of a Geospatial Semantic Web-Based Spatial Decision Support System for Digital Earth." *International Journal of Digital Earth* 3 (2): 111–134. doi:10.1080/17538940903373803.

Zhao, P., T. Foerster, and P. Yue. 2012. "The Geoprocessing Web." *Computers & Geosciences* 47: 3–12. doi:10.1016/j.cageo.2012.04.021.