A Study on Data Processing Services for the Operation of Geo-Analysis Models in the Open Web Environment

Article · November 2018

DOI: 10.1029/2018EA000459

6 authors, including:

Min Chen
Key Laboratory of Virtual Geographic Environment, Ministry of Education of PRC
78 PUBLICATIONS **689** CITATIONS

SEE PROFILE

Guonian Lü
Key Laboratory of Virtual Geographic Environment, Ministry of Education of PRC, …
147 PUBLICATIONS **859** CITATIONS

SEE PROFILE

Songshan Yue
Nanjing Normal University
21 PUBLICATIONS **115** CITATIONS

SEE PROFILE

Yongning Wen
Nanjing Normal University
55 PUBLICATIONS **313** CITATIONS

SEE PROFILE

How to cite this article: Wang J, Chen M, Lü G, Yue S, Chen K, Wen Y. 2018. A Study on Data Processing Services for the Operation of Geo-Analysis Models in the Open Web Environment. Earth and Space Science, Doi: https://doi.org/10.1029/2018EA000459

A Study on Data Processing Services for the Operation of Geo-Analysis Models in the Open Web Environment

Jin Wang[1, 3, 4], Min Chen[1, 2, 3, *], Guonian Lü[1, 3, 4], Songshan Yue[1, 3, 4], Kun Chen[1, 3, 4], Yongning Wen[1, 3, 4]

[1] Key Laboratory of the Virtual Geographic Environment, Ministry of Education, Nanjing Normal University, Nanjing, Jiangsu, China.
[2] Key Laboratory of Watershed Ecology and Geographic Environment Monitoring, NASG, Nanchang, Jiangxi, China.
[3] Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing Normal University, Nanjing, Jiangsu, China.
[4] State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing Normal University, Nanjing, Jiangsu, China.

Corresponding author: Min Chen (chenmin0902@163.com)

Key Points:

- Data preparation and preprocessing for model driving
- Data processing service for coupling models with related data resources
- Model integration with data processing services for comprehensive problem solving

Abstract

With the development of network technology, integrated modeling frameworks based on web services are becoming a key topic with regard to solving complex geographic problems. Currently, large numbers of geo-analysis models and massive data resources are available on the open web. Accessing, acquiring or invoking individual resources transparently is relatively straightforward; but combining these models and data resources for comprehensive simulations remains challenging due to their heterogeneity and diversity. Data resources are the driving force of model execution and can serve as an intermediate linkage medium for model integration, yet in most cases, the data resources cannot be directly used to drive or link models. Data processing services that can prepare and process data are urgently needed in the web environment to enable the convenient coupling of models and data resources through the web, thus reducing the difficulty of preparing data and avoiding repetitive data processing work. Three types of data processing methods that can achieve mapping, refactoring, and visualization are designed in this article based on the proposed Universal Data eXchange (UDX) model. These methods can be published as data processing services to facilitate the operation of geo-analysis models through the web environment. The applicability of the proposed data processing services is examined through two cases: the processing service is applied in model integration through the web in the first case, and data preparation with processing services for the Taihu Water Pollution Diffusion Model (TWPDM) is carried out in the second case. The results demonstrate that the proposed processing services can bridge the gap between geo-analysis models and data resources hosted on networks.

1 Introduction

Geographic modeling is an effective way to reveal the nature of geographic phenomena (Demeritt and Wainwright, 2005). Geo-analysis models are the products of geographic modeling and are employed for geographic analysis (Lü, 2011; Wen et al., 2016; Yue et al., 2016). A geo-analysis model is an abstraction of the evolution of geospatial entities and their relationships (Belete et al., 2017a). The use of geo-analysis models enables the simulation and reproduction of geographic processes to make decisions and explore geographic rules. When considering complex geographic problems, typically, geo-analysis models must be integrated to solve complex geographic problems due to the limited capacity of a single model (Overeem et al., 2013; Jones et al., 2014; Guzman et al., 2015; Rajib et al.,2016; Belete et al., 2017b; de Bakker et al., 2017; Buahin et al., 2018; Marcot et al., 2018; Rossetto et al., 2018; Tian et al., 2018). After years of development, massive numbers of geo-analysis models in multiple disciplines and domains have been produced all over the world (Goodchild, 1996; Goodall et al., 2010; Jagers, 2010; Voinov et al., 2010; Granell et al., 2013b; Laniak et al., 2013; Yue et al., 2016). With the large number of geo-analysis models available, the development of methods for adequately integrating these models, building synthetic geographic simulation systems, and solving complex geographic problems has gradually become an important topic (Díaz L et al., 2008, Granell et al., 2013a). As information technology (IT) has developed, progress has been made in terms of studying integrated modeling frameworks. The development of these frameworks has passed through approximately three stages. (1) In the hard-coding stage, hard coding is used to directly integrate models. Although this method for model integration is direct and explicit, it lacks flexibility, and the work undertaken for processing data (i.e., data preprocessing and data conversion) is difficult to reuse due to the diversity of data requirements among geo-analysis models (Granell et al., 2013b; Belete et al., 2017a). (2) In the component-based modeling and simulation stage, the use of component-based strategies, in which independent components or modules are designed, reduces the dependencies among components and frameworks. However, these strategies are limited to specific modeling frameworks, and the components are difficult to reuse in other frameworks. Additionally, these components cannot be used directly on networks (Argent et al., 2004, 2006; Jagers, 2010; Granell et al., 2013b). (3) In the web-service-based modeling and simulation stage, web-service technologies are used to publish geo-analysis models and related data as services (Hull et al., 2006; Wen et al., 2006; Granell et al., 2010, 2013a; Zhao et al., 2012; Yin et al., 2015; Jones et al., 2016; Belete et al., 2017a). Therefore, these resources can be reused to support the collaborative solving of complex geographic problems. Regarding this aspect, the Open Geospatial Consortium (OGC) has proposed several service description specifications such as the Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS) and Web Processing Service (WPS) implementation standards (Botts et al., 2008; Reichardt et al., 2010; Granell et al., 2014). These service specifications are widely used in the field of geographic modeling and model-related data (i.e., the raw data related to models, including input data, output data and control parameters) sharing in the open web environment; sample applications include the 52° North project (http://52north.org/) and Quantum GIS (QGIS) (QGIS Development Team, 2012). Moreover, the Simple Object Access Protocol (SOAP), the Web Services Description Language (WSDL) and cloud computing are widely used in the sharing and integration of geo-analysis models (Geller & Turner, 2007; Goodall et al., 2011, 2013; Yang et al., 2011; Wen et al., 2013, 2017; David et al., 2014; Belete et al., 2017a, 2017b; Jin et al., 2017).

Considering the development of integrated modeling frameworks, the concepts of sharing and reusability, which are based on web-service technologies, are widely used for sharing model and data resources. However, using existing data resources to drive models that run in the open web environment remains difficult challenging. On the one hand, preparing data for use with models is difficult due to the heterogeneity and diversity of the input and output data related to geo-analysis models. Although massive data resources exist on the web, many of those data still cannot be used directly. On the other hand, traditional data processing tools typically exist as independent software modules or algorithms that run on desktop computers, and many of them cannot be directly used through the web. Examples include the data processing tools in the ArcGIS Toolbox (a component in the ArcGIS software, which is provided by the Environmental Systems Research Institute, ESRI) and the data processing subprogram in the Community Surface Dynamics Modeling System (CSDMS) (Peckham et al., 2013) model repository. Avoiding the issues that arise from the heterogeneity and diversity of data resources and the reuse of data processing methods has become a key topic in current model integration studies.

The Open Geographic Modeling and Simulation System (OpenGMS) team has constructed the OpenGMS platform (http://geomodeling.njnu.edu.cn/) to enable the development of solutions for complex geographic problems (Chen et al., 2009a, 2009b, 2011, 2015, 2017; Wen et al., 2013, 2017; Yue et al., 2015, 2016; Lü et al., 2017). The OpenGMS platform has drafted several standards and specifications (i.e., model-related metadata and classification) for geo-analysis model descriptions and aims to make good use of geo-analysis models, data resources and computing resources that are distributed over the web. This platform represents an attempt to develop a series of methods and technological solutions to enable collaborative modeling via geographic conceptual modeling, sharing geo-analysis models, and integrating and simulating geographic processes in the open web environment. Among the many research areas of interest in the platform, linking models and data resources is a key issue in the implementation of the integration and simulation of geographic processes on networks.

Within the OpenGMS research framework, this article presents the development of data processing services with the aim of addressing the gap between models and data resources. The study is based on the Universal Data eXchange (UDX) model, which was also designed by the OpenGMS team. Previous research has focused on the concept of the UDX model and the expression (e.g., data content, semantic information, units and dimensions) of model-related data with the UDX model (Yue et al., 2015, 2016). This article follows previous research and studies resource (model and data resources) integration with the UDX model in the open web environment, and methods for designing, generating, managing, publishing, and invoking data processing services based on the UDX model are then designed to form a series of shareable and reusable services for the mapping, refactoring, and visualization of data. The data processing services proposed in this article will reduce the difficulties encountered by model users when applying geo-analysis models.

The remainder of this article is structured as follows. The basic application scenario and a conceptual classification of data processing services are explained in Section 2. In Section 3, the methods used in implementing services, including those that map, refactor and visualize data, are introduced. Section 4 presents the design of the data service container that can be used to share and reuse data processing services. Section 5 examines the applicability of the proposed data processing service by using two case studies. Finally, a discussion of the study and the conclusions are presented in Section 6.

2 Basic application scenario and conceptual classification of data processing services

Due to the heterogeneity and diversity of geo-analysis models, the data associated with these models are typically complex, and such data are diverse in terms of their forms and contents. These features cause difficulties when integrating and applying models. Implementing services for data processing when operating models on distributed networks has been an inevitable tendency.

In this section, a case of model integration is introduced to illustrate data processing services and how they work (Figure 1).

Figure 1 shows a model integration scenario of suitable area selection for planting apple trees. The suitable area should meet the following four requirements. (1) The area should be sunny. (2) The trees should be mainly distributed on both sides of a valley within 500 m. (3) The temperature should between 10 and 12 degrees centigrade. (4) Rainfall should between 550 and 680 mm. As shown in Figure 1, four specific areas were selected according to these four requirements (referred to four workflows with different colors in the figure). Then, an overlay analysis (“Model E”) is conducted based on these four areas to generate a comprehensive result. The raw input data in this case include raw DEM (digital elevation model) data and data for the temperature and rainfall sampling points in the study area. The raw DEM data provide the raw elevation information and are used to calculate the sunny slope and extract the river network. The data for the temperature and rainfall sampling points are used for the interpolation and extraction of eligible subsets. Finally, all the outputs from previous calculations (“Model A-D”) are used to conduct the overlay analysis (“Model E”). The output of the overlay analysis is the suitable areas that meet all the requirements for planting apple trees. In this case, the raw DEM data were ASCII GRID (an ASCII-based raster data format developed by ESRI), and the data structure of the temperature and rainfall sampling points were customized by the model developers of “Model C” and “Model D.” The “data processing” in this case involves data conversion between two models. For example, the output of “Model A” was shapefile (a binary data format developed by ESRI), while the input of “Model E” was GeoTIFF; thus, the “Shapefile to GeoTIFF” tool is used to convert shapefile to GeoTIFF.

In this case, several models are involved. To select a suitable planting area, these models should be linked together to form a model integration scenario, as shown in Figure 1. Preparing data and running the model integration scenario are not easy for those who are not familiar with these models. For example, some researchers may be unfamiliar with GIS and the concepts of vectors, rasters, DEMs, etc., and may not understand ASCII GRID, GeoTIFF, or shapefiles. Furthermore, the same data may be understood differently in the context of different research fields. These issues limit the use of models, as well as their communication and integration in multidisciplinary research. Therefore, reducing the costs of using cross-domain models is a key point in interdisciplinary studies.

If model users want to use a model, they must first understand the input/output data of the model. However, it may be difficult for users to understand unfamiliar data formats, especially the customized data of the model. In this case, model users should invest considerable effort to understand the data formats of the model. Additionally, these works cannot be reused by others. Therefore, we proposed the UDX model to describe model-related data in a flexible and understandable manner to help model users more clearly understand the data requirements of models (Wen et al., 2013; Yue et al., 2015). With the UDX model, model users can focus on data value of model-related raw data rather than the complex organizational structure of the raw data. Theoretically, they can drive the model as long as they provide related data, and their data processing-related work can also be made available to others.

The UDX model provides a structured strategy to represent heterogeneous data and includes a series of standard data operation interfaces to enable data expression. With the UDX model, model providers can describe model-related data interfaces unambiguously, and model users who are not familiar with these models can easily understand model-related data (Yue et al., 2016; Zhu et al., 2017). The UDX model includes two components: UDX data and UDX schema. The UDX data express the actual value of the data, whereas the UDX schema contains a description of the data. Figure 2 shows how the UDX model describes the raw data (i.e., ASCII GRID raw data). The left panel in Figure 2 (a) shows the ASCII GRID raw data, which include elevation information and related metadata. These contents of the raw data can be organized into a structured expression by the UDX model, as shown in the right panel. The elevation information and metadata are expressed as data nodes with strict data type requirements. In Figure 2 (b), the upper left panel shows the UDX data of the ASCII GRID data (i.e., the UDX data), whereas the other panel shows a description of the raw data (i.e., the UDX schema). Obviously, the UDX data of the ASCII GRID are more user friendly: the model user does not have to understand the raw data format as long as he/she understands the corresponding UDX data. Then, the model user can prepare the data for the model according to the UDX schema that describes the model data.

UDX data can clearly express the model-related data to model users, but if these data are expressed by the UDX model, such as the data in Figure 1, certain problems still need to be solved.

First, the model should understand and run with the UDX data. Therefore, a data mapping method must be designed to transform UDX data into raw data. A data mapping method will provide a channel for data exchange between the UDX data and raw data. As shown in Figure 1, “Model B” requires raw ASCII GRID data as input, but the given data may be UDX data prepared by model users. The related data mapping method should be invoked to transform the UDX data to ASCII GRID raw data before running the model.

Second, the UDX data prepared by model users may not always match the model input requirements. For example, “Model A” requires GeoTIFF UDX data as input, but the provided UDX data are ASCII GRID. Thus, a data refactoring method must be designed to convert the data expressed by different UDX structures, such as ASCII GRID UDX data and GeoTIFF UDX data. Namely, data refactoring will provide methods for converting one type of UDX data into another.

Third, data visualization will contribute to data preparation and result validation. Therefore, a data visualization method must be designed to examine the reasonableness of the data from a visual perspective. For example, the reasonableness of the extracted river network (in “Model A”) can be assessed in a visual way.

In summary, data processing methods (i.e., the mapping method, refactoring method and visualization method) based on the UDX model can reduce the cost of using models. Although developing these data processing methods is tedious and complex, once these data processing methods can be published as services, they can be reused to avoid repetitive work during complex data preparation tasks, especially for those who are not familiar with complicated data.

To publish these data processing methods as services, there is still a need to design a data service container to publish and manage the services in the open web environment so that the model user can then invoke the processing service to prepare the data rather than manually preparing the data after understanding the raw data related to the model.

Figure 3 explains the functionality and classification of data processing services based on the UDX model, the corresponding implementation strategies, the sharing of services, and application scenarios.

Based on these data processing services, an application scenario of using models can be described as follows. Model providers encapsulate the data interfaces of their models by using standard encapsulation interfaces and publish the encapsulated models as model services in the open web environment (Yue et al., 2016). The data processing methods are developed by the person who best understands the model-related data. Then, these methods can be published as services and invoked to process data to run a model.

3 Implementing service methods

3.1 Data mapping method

Before executing a model, the model-related data requirements (the form in which the data are expressed, organization of the contents of the data, semantic information and other features) should be clearly understood by model users. Among these features, the data format can be determined to determine the expression of the data. In practice, no universal data format can be used to express all types of data because of the complexity and comprehensive nature of geographic modeling. The data formats that are commonly used in the field of geographic modeling are listed below.

(1) Custom data formats. The format of model-related data is typically determined by the programming habits of the model developer. Examples include custom text files, binary files and image files. In specific geo-analysis models, the custom data read-write interface must be completely consistent with the organization of the custom data format. For example, in the previous case, the data of the raw temperature or rainfall sampling points include X/Y coordinates and the temperature or rainfall values at the position. The data content can be organized into multiple formats, such as text files or binary files. The text files include .csv (comma-separated values, CSV) files, in which the coordinates and the values can be separated by commas, and .txt files, in which the separator can be any character (space, semicolon, etc.) determined by the model developer. The binary files include custom binary files and other common data formats, such as shapefiles. Although the same data contents can be organized into different data formats, the data read-write interface is determined once the data format is selected.

(2) Domain-specific data formats. With the development of geography, several widely used data formats have been accumulated and have become domain-specific data formats. Developers of geo-analysis models also employ mature open-source or commercial data formats, including ASCII GRID files, shapefiles, GeoTIFF images, and NetCDF files (Rew and Davis, 1990). Existing data read-write interfaces and APIs are typically used to read/write data in domain-specific data formats. The Geospatial Data Abstraction Library/OGR Simple Features Library (GDAL/OGR) (Fundation O. S. G., 2008) is among

the most commonly used data read-write libraries and supports several raster data formats such as Arc/Info Binary Grid (.adf), Arc/Info ASCII Grid (.asc), NetCDF (.nc), GeoTIFF (.tif) and Erdas Imagine (.img).

(3) Computer memory data formats. Generally, this type of data format exists as a variable in computer memory when a program is running. Such data formats typically contain the parameters for primary functions (generally the main function) and the parameter list for API functions. These types of data formats are usually expressed by simple data types in the programming language under use, such as int, float, double, or string.

In summary, multitudinous data formats are used in modeling, and model users have difficulty studying all data formats to prepare data for running models. To reduce the costs of preparing data prior to running a model, this article presents an extensible data mapping method based on the UDX model. The implementation details of the data mapping method are shown in Figure 4. The raw data related to the models can be organized into a single data file or multiple data files. Regardless of whether the raw data are expressed in a custom data format, a domain-specific data format or another format, several corresponding read-write interfaces can be used to access them. As shown in the upper-right panel, several read-write interfaces exist to access raw data, and these APIs can be implemented in multiple ways, including Dynamic Link Libraries (DLLs), executable programs (EXE files), and Component Object Model (COM) files; GDAL/OGR is an open-source library that provides read-write interfaces for raster or vector data in the domain of spatial information. The standard Object Linking and Embedding Database (OLE-DB) and Open Database Connectivity (ODBC) libraries are commonly used to read/write raw data in databases. The data mapping method exposes two interfaces to operate UDX data nodes: one reads the data content from the UDX data (ReadFromNode), and the other writes the data content to the UDX data node (WriteToNode). The two interfaces exchange information with the raw data by accessing the raw data read/write interfaces, as shown in the lower-right panel in Figure 4.

Figure 5 shows the pseudocode for exchanging information between shapefile raw data and the corresponding UDX data. Here, the GDAL/OGR library is used to read and write raw shapefile data and the UDX model provides a series of interfaces for UDX node operation. In Figure 5 (a), the interface “ReadFromNode” reads the data content from the UDX data node and organizes this information as a shapefile by using the GDAL/OGR library. In Figure 5 (b), the interface “WriteToNode” reads the data from the shapefile raw data and writes them to the UDX data node. The case in Section 2 involves the mapping methods for ASCII GRID, shapefile, GeoTIFF and customized data (e.g., the data for the temperature and rainfall sampling points and the control parameters).

3.2 Data refactoring method

Data refactoring involves reprocessing data to fit the requirements of the target model; the target model can be executed by using the refactored data. The refactoring method is based on the UDX model and exchanges data between different UDX data formats. Data refactoring can be classified into the following two categories.

(1) Functional refactoring. This refactoring method processes the values of UDX data nodes; it does not change the structure of the UDX nodes. For example, when transforming the coordinates in UDX data to another projection, only the data value of the UDX node changes; the UDX node structure does not change. Spatial analysis algorithms that are commonly used in the field of GIS, including raster data calculation and clipping algorithms, are typically employed in the functional refactoring method. Open-source libraries are also used in this type of data refactoring, including the Gmsh (Geuzaine and Remacle, 2009), proj4 (Urbanek, 2008) and GDAL libraries, which perform mesh subdivision, projection transformations, and raster data reading/writing, respectively.

(2) UDX node structure refactoring. Frequently, the same data content can be organized into different UDX node structures; thus, the refactoring method is used to reorganize one type of UDX node structure into the target structure. Figure 6 shows the refactoring of a UDX node structure from ASCII GRID UDX data to GeoTIFF UDX data. The ASCII GRID UDX data include a header node and a body node. The header node includes the NCOL node, which expresses the width of the grid; the NROW node, which expresses the height of the grid; the Xllcorner and Yllcorner nodes, which express the upper left position of the grid; and so on. The body node is a collection node that collects the rows of the grid; each row is an array of elevations. The GeoTIFF UDX structure also includes a header node and a body node. The NCOL, NROW, XllCorner, YllCorner, CellSize, and body nodes of ASCII GRID data correspond to the Width, Height, Upper left corner X, Upper left corner Y, Pixel Width (Pixel Height), and Value nodes of GeoTIFF data, respectively.

In summary, data refactoring includes the following operations: build a UDX node structure, read/write the UDX node value and perform numerical calculations. To efficiently implement the specific requirements, third-party libraries such as Gmsh, proj4 and GDAL can be used in the refactoring method. In short, the means of implementing the refactoring method are very flexible, and any desired UDX data structure can be built by using UDX node operation interfaces. The case in Section 2 involves two types of refactoring methods: one is an ASCII GRID refactor to GeoTIFF, and the other is a shapefile refactor to GeoTIFF (i.e., converting the vector data to raster data). These two refactoring methods are used to dynamically exchange data during the integrated scenario.

3.3 Data visualization method

When a geo-analysis model completes its execution, the model output is displayed to permit a visual assessment of the correctness of its execution. As web-based technologies have been developed, large numbers of mature data visualization engines have emerged. Engines that are used to perform Earth and map visualizations include WorldWind (a free and open-source API for virtual globes, https://worldwind.arc.nasa.gov/), Cesium (an open-source JavaScript library for world-class 3D globes and maps, https://cesiumjs.org/), OpenLayers (a high-performance, feature-packed library for all mapping needs, https://openlayers.org/), and Three.js (an open-source, easy-to-use and lightweight 3D library, https://threejs.org/). D3.js (a JavaScript library for manipulating documents based on data, https://d3js.org/), Highcharts (a JavaScript charting framework that enables the easy production of interactive charts for webpages, https://www.highcharts.com/), and Echarts (a powerful, interactive charting and visualization library for browsers, http://echarts.baidu.com/) are used for chart visualization. Other visualization engines that serve other purposes also exist. These visualization engines provide many excellent functions or APIs that can display data in various ways; however, enabling these engines to support a specific data expression or provide compatibility for all types of data formats is difficult. To make a new data format compatible, the new data format must be converted to a form that is supported by that engine. Moreover, sharing and reusing these conversion methods is difficult because of differences among these visualization engines.

Generally, development of a common data visualization engine that can represent all types of data formats is difficult, and new data formats are continually emerging in the field of geographic information science. The concept of a visual package is proposed in this article to support compatible new data formats and flexibly display data with specific styles. When new data formats appear, new visual packages will be developed, and specific visual styles can be customized in those packages. The visual package is also based on the UDX model.

Figure 7 shows the details of a visual package. Within the package, third-party visualization engines are employed as renderers (Cesium.js for Earth visualization, OpenLayers for map visualization, Echarts for chart visualization, and so on); UDX data processing, which includes reading the UDX data nodes, configuring the UDX nodes and binding the visualization engine based on these renderers, is then performed. Three main interfaces (getUdxSchema, getUdx, and getSchemaIndex) are exposed outside the package. The “getUdxSchema” interface enables acquisition of the UDX data structure. The “getUdx” interface obtains the UDX data outside the package. If a visual package supports multiple types of UDX data formats, then the “getSchemaIndex” interface refers to the specific UDX data format that will be resolved by the package. With this loosely coupled interface design, the visual package can be migrated to any other service platform after the exposed interfaces have been implemented.

When using visual packages, UDX data can be flexibly displayed and analyzed in multiple styles and even display portions of the UDX data as long as the visual package supports the extracted UDX data. This approach is very useful for displaying portions of model output data when the output data are too numerous to be displayed in their entirety.

4 Design of a data service container for data processing services

To enable sharing and reuse of the proposed data processing methods over distributed networks, a data service container has been designed. This container is mainly used to publish services and provides interfaces to manage and invoke them.

Figure 8 shows the architecture of the data service container. The data service container takes Node.js as its main development language due to its lightweight features and efficiency. MongoDB is used to store massive model-related data because it shows excellent performance when used to store and process large datasets. Massive data can be efficiently queried and updated when using Node.js and MongoDB, and the performance of the data service container is guaranteed when processing large datasets. The HTTP protocol is employed as the communications protocol in the data service container. This protocol supports service management, service publishing and invoking, and other functions.

Before data processing methods can be published as services, they should be bundled into a package for upload to a data service container. Taking the mapping service as an example, the mapping method can be developed by using any programming language (e.g., C/C++, C#, Java, Fortran, etc.) and can be any type of program (e.g., .EXE, .DLL, .LIB) as long as the program complies with the invocation interfaces designed by the data service container. Then, the program of the mapping method, the description document, the UDX schema indicating the supported type of UDX data and other related resources are packaged into a .zip file. This .zip file can be uploaded to the data service container and then published as a mapping service in the open web environment.

The data service container is a host of data processing services and an avenue for service invocation. This container maintains all the information of the service, such as the service list (e.g., service detail information, author information, and publication date), the invocation information (e.g., invoker information, invocation time and the status of invocation), service invocation records (e.g., service invocation log information, invocation exceptions, and input/output data), and the service status information (e.g., availability status). In addition to managing the data processing services, the data service container can manage users and related data resources. User management mainly applies to local and remote users and includes management of user accounts and simple permissions. Data management mainly refers to management of data that a user uploads to the data service container to invoke the data processing service and the data that the data processing service outputs after its execution.

Invocation of the data processing service is asynchronous. The Node.js application starts a new thread for each new invocation of a service, and access to and invocation of other services are therefore unaffected. A service-operation instance is generated by each invocation and is used to trace the running status of the service (such as whether it has thrown an exception or whether the output files have been generated). Once an invocation is finished, an invocation record is generated to record the information of the invocation.

The data service container can be installed in different server nodes, and the nodes can communicate with one another. Specifically, a container on a specific server node can access and invoke data processing services published by other containers. Therefore, everyone can access the data processing service published by others in the open web environment.

5 Case studies

Two cases are presented in this section to examine the applicability of the proposed data processing service (demos of the two cases can be accessed from http://ogms.gitee.io/opengms/). The first case, whose application scenario is introduced in the second section, mainly shows how data processing services dynamically exchange data during model integration. The second case focuses on application of the data processing service for data preparation.

5.1 Data processing service in the model integration scenario

In this section, a scenario in which planting areas are selected for apple trees, introduced in Section 2, demonstrates the applicability of the proposed data processing service to the process of model integration. As shown in Figure 9 (a), the model integration scenario was implemented on the model integration platform (developed by OpenGMS team). In this scenario, all the data processing services and model services were linked together for a specific computational task. Three refactoring services dynamically exchange the data during model invocation; models A-D are the implementations of requirements 1-4 (noted in Section 2), and model E is for calculating the final result by overlaying the output of models A-D. In Figure 9 (b), the intermediate output data and the final result data are displayed by the visualization service. The center of the panel shows the final suitable areas for planting apple trees, and the remaining four panels show the middle results that meet requirements 1-4.

5.2 Data preparation by using data processing services

This case study examines the applicability of the proposed data processing service to prepare the data for the Taihu Water Pollution Diffusion Model (TWPDM) (Zheng S., 2016). The TWPDM focuses on the simulation of water pollution diffusion for China’s Taihu Lake. The framework of the TWPDM is presented in Figure 10. The input of the model mainly includes wind data, pollution data, data from the inlet rivers of Lake Taihu (i.e., inlet position, river name, river ID, etc.), initial pollutant concentration data, and other model control parameters. After calculation, the output of the model includes the pollutant concentration in every specific moment, the water depth, and the average water velocity. The input or output data are all customized data as determined by the model developer. In this case, the typical input data (wind data, which are marked in red in Figure 10) are selected to illustrate how the data processing service prepares the data for model invocation. Finally, a data visualization service is invoked to display the scenario of pollution diffusion.

The TWPDM was encapsulated and published to the model server as a model service. The mapping methods for the raw wind station data were published as a service in the data service container. The wind station data requirements were described by the UDX model, as shown in Figure 11 (a). The wind station data included the following information: the station id, the station name, the id of the grid to which the station belonged, and the position of the station. Users can easily understand the UDX description of these wind stations. To drive the model, the model user needs only to prepare the wind station UDX data, as shown in Figure 11 (b); he/she does not need to understand the details of the raw wind station data. The mapping service can convert the wind station UDX data to raw data to correctly drive the model. However, the raw wind station data can be organized into various data formats because of the specific data-collection environment, as shown in Figure 11 (c). Two types of raw wind station data formats exist (i.e., .txt file and .csv file), both of which include the required wind station data. Either of these data formats can generate UDX data to drive the model as long as the corresponding mapping service has been published. Similarly, manually generating the simple UDX data is effective for the simple raw data, such as by preparing model control parameters.

The wind data included the wind velocity and direction at each wind station and the time steps. The raw data organized this information into a .txt file. In Figure 12, “UDX schema A” shows the UDX description of the raw data in which the wind data were organized by their time steps, namely, data for all stations at the first moment, data at the second moment, etc. However, the wind data required by the TWPDM were organized into two .txt files: one stored the wind velocity and direction, and the other stored the time steps. In Figure 12, “UDX schema B” and “UDX schema C” show the UDX descriptions of the two files. The wind data in “UDX schema B” were organized by the wind stations, namely, data for all the time steps at the first wind station, data at the second station, etc. Although “UDX schema A” provided all the required wind data for the model, the model required two UDX subsets, which are expressed by “UDX schema B” and “UDX schema C.” Therefore, “UDX schema A” must be refactored as “UDX schema B” and “UDX schema C” by converting one file to two. Meanwhile, the structure of the wind data is reorganized to meet the data requirements of TWPDM.

The other data preparation method was similar to that for the wind station data and wind data. Figure 13 shows the interface of TWPDM invocation. First, the related data processing services and the TWPDM model service were selected from the left panel to build the model integration scenario. Then, these services were linked to build a workflow from data preparation to model invocation. After the model is finished, the visualization service can be invoked to simulate the scenario of pollutant concentrations in Taihu Lake. Following the above steps, the published data processing services can be shared and reused for other TWPDM model users.

6 Conclusions and future work

The purpose of this study is to provide a method for linking geo-analysis models and their related data resources. With the proposed data processing service, the data resources can be processed to drive models in the open web environment. The processing service is the medium for model integration, not only providing a simple means of access to geo-analysis models but also reducing the repetitive work of data preparation through using these services. Finally, the study of data processing services provides new concepts in the study of integrated modeling frameworks.

As the study of geo-modeling and the integration of geo-analysis models continues, certain avenues should be explored further.

(1) Transmission of UDX data in networks. In general, geo-analysis model-related data are “big data” and are difficult to transmit over networks. Therefore, appropriate progressive transmission strategies must be explored, including data compression, data transmission in multiple batches and new transmission strategies in the IT domain.

(2) Automated tools for generating data processing methods easily. Currently, data processing methods are always generated by coding. As the data related to models are generally complex, coding for processing these data is tedious and fallible. Therefore, easy-to-use and more highly automated tools are needed to improve the efficiency of generating data processing methods. Additionally, more people would partake in this work if user-friendly tools were available.

(3) Control of user permissions for accessing data. Generally, certain geo-analysis models and their related data are sensitive, or in other cases, the model provider wants to

maintain control over his/her model. Therefore, strategies for managing user permissions and security controls should be designed to guarantee the security of the resources in the open web environment.

Acknowledgments, Samples, and Data

We appreciate the detailed suggestions and comments from the secretariat and the anonymous reviewers. We express heartfelt thanks to the other members of the OpenGMS team. This work was supported by the NSF for Excellent Young Scholars of China under Grant number 41622108, the National Basic Research Program of China (973 Program) under Grant number 2015CB954103, and Priority Academic Program Development of Jiangsu Higher Education Institutions under Grant 164320H116. The samples and data related to this article can be found in: http://ogms.gitee.io/opengms/.

References

52° North Initiative for Geospatial Open Source Software GmbH, Münster, Germany. http://52north.org/ (accessed 01.05.18).

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

Argent, R. M., Voinov, A., Maxwell, T., et al. (2006), Comparing modelling frameworks–a workshop approach. *Environmental Modelling & Software*, 21(7), 895-910.

Belete, G. F., Voinov, A., Laniak, G. F. (2017a), An overview of the model integration process: From pre-integration assessment to testing. *Environmental Modelling & Software*, 87, 49-63.

Belete, G. F., Voinov, A., Morales, J. (2017b), Designing the Distributed Model Integration Framework–DMIF. *Environmental Modelling & Software*, 94, 112-126.

Botts, M., Percivall, G., Reed, C., et al. (2008), OGC? sensor web enablement: Overview and high level architecture. *GeoSensor networks*, 175-190.

Buahin, C. A., Horsburgh, J. S., (2018), Advancing the Open Modeling Interface (OpenMI) for integrated water resources modeling. *Environmental Modelling & Software*, 108, 133-153.

Chen, M., Lin, H., Liu, D., et al. (2015), An object-oriented data model built for blind navigation in outdoor space. *Applied Geography*, 60, 84-94.

Chen, M., Lin, H., Lü, G. (2017), Virtual Geographic Environments. *The International Encyclopedia of Geography*, 1–11.

Chen, M., Sheng, Y., Wen, Y., Su, H. (2009a), Geographic Problem-Solving Oriented Data Representation Model. *Journal of Geo-Information Science*, 11, 333-337.

Chen, M., Sheng, Y., Wen, Y., Tao, H., Guo, F. (2009b), Semantics guided geographic conceptual modeling environment based on icons. *Geographical Research*, 28, 705-715.

Chen, M., Tao, H., Lin, H., et al. (2011), A visualization method for geographic conceptual modelling. *Annals of GIS*, 17, 15-29.

David, O., Lloyd, W., Rojas, K., et al. (2014), Model-as-a-service (MaaS) using the cloud services innovation platform (CSIP).

Díaz L. , Granell C., Gould M. (2008), Case Study: Geospatial Processing Services for Web based *Hydrological Applications*.

De Bakker, M. P., De Jong, K., Schmitz, O., et al. (2017), Design and demonstration of a data model to integrate agent-based and field-based modelling. *Environmental Modelling & Software*, 89, 172-189.

Demeritt, D., & Wainwright, J. (2005), Models, modelling and geography. *In: N. Castree, A. Rodgers and D. Sherman, eds. Questioning geography. Oxford: Blackwell*, 206–225.

Fundation O. S. G. (2008), GDAL-OGR: Geospatial Data Abstraction Library/Simple Features Library Software.

Geller, G. N., Turner, W. (2007), The model web: a concept for ecological forecasting. *Geoscience and Remote Sensing Symposium*, IGARSS, 2469-2472.

Geuzaine, C., Remacle, J. F. (2009), Gmsh: A 3- D finite element mesh generator with built- in pre- and post- processing facilities. *International journal for numerical methods in engineering*, 79, 1309-1331.

Goodall, J. L., Castronova, A. M., Elag, M., et al. (2010), An integrated modeling environment within the CUAHSI Hydrologic Information System. *AGU Fall Meeting Abstracts*.

Goodall, J. L., Castronova, A. M., Huynh, N. N., et al. (2013), Using a Service-Oriented Approach to Simulate Integrated Urban Infrastructure Systems. *Journal of Computing in Civil Engineering*, 29(5), 04014061.

Goodall, J. L., Robinson, B. F., Castronova A. M. (2011), Modeling water resource systems using a service-oriented computing paradigm. *Environmental Modelling & Software*, 26(5), 573-582.

Goodchild, M. F. (1996), GIS and environmental modeling: progress and research issues. *John Wiley & Sons*.

Granell C., Díaz L, Schade S, et al., (2013a), Enhancing integrated environmental modelling by designing resource-oriented interfaces. *Environmental Modelling & Software*, 39, 229-246.

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

Granell, C., Díaz, L., Tamayo, A., et al. (2014), Assessment of OGC web processing services for REST principles. *International Journal of Data Mining, Modelling and Management*, 6, 391-412.

Granell, C., Schade, S., Ostl?nder, N. (2013b), Seeing the forest through the trees: a review of integrated environmental modelling tools. *Computers, Environment and Urban Systems*, 41, 136-150.

Guzman, J. A., Moriasi, D. N., Gowda, P. H., et al. (2015), A model integration framework for linking SWAT and MODFLOW. *Environmental Modelling & Software*, 73, 103-116.

Hull, D., Wolstencroft, K., Stevens, R., et al. (2006), Taverna: a tool for building and running workflows of services. *Nucleic acids research*, 34 (suppl_2), W729-W732.

Jagers, H. R. A. (2010). Linking data, models and tools: an overview.

Jin, X., Robinson, K., Lee, A., et al. (2017), A prototype cloud-based reproducible data analysis and visualization platform for outputs of agent-based models. *Environmental Modelling & Software*, 96, 172-180.

Laniak, G. F., Olchin, G., Goodall, J., et al. (2013), Integrated environmental modeling: a vision and roadmap for the future. *Environmental Modelling & Software*, 39, 3-23.

Jones, A. S., Horsburgh, J. S., Jackson, S. D., et al. (2016), A web-based, interactive visualization tool for social environmental survey data. *Environmental Modelling & Software*, 84, 412-426.

Jones, N., Nelson, J., Swain, N., et al. (2014), Tethys: a software framework for web-based modeling and decision support applications.

Lü, G. N. (2011), Geographic analysis-oriented virtual geographic environment: framework, structure and functions. *Science China (D)*, 54(5):733–743

Lü, G., Chen, M., Yuan, L., et al. (2017), Geographic scenario: a possible foundation for further development of virtual geographic environments. *International Journal of Digital Earth*, 1-13.

Marcot, B. G., Penman, T. D., (2018), Advances in Bayesian network modelling: Integration of modelling technologies. *Environmental Modelling & Software*.

Mohammad, A. R., Venkatesh, M., I, L. K., et al. (2016), SWATShare – A web platform for collaborative research and education through online sharing, simulation and visualization of SWAT models. *Environmental Modelling & Software*, 75, 498-512.

Overeem, I., Berlin, M. M., Syvitski, J. P., (2013), Strategies for integrated modeling: The community surface dynamics modeling system example. *Environmental modelling & software*, 39, 314-321.

Peckham, S. D., Hutton, E. W. H., Norris, B. (2013), A component-based approach to integrated modeling in the geosciences: The design of CSDMS. *Computers & Geosciences*, 53, 3-12.

QGIS Development Team. (2012), “QGIS Geographic Information System.” Open Source Geospatial Foundation Project.

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.

Rew, R., Davis, G. (1990), NetCDF: an interface for scientific data access. *IEEE computer graphics and applications*, 10, 76-82.

Ritter, N., Ruth, M., Grissom, B. B., et al. (1995), GeoTIFF format specification GeoTIFF revision 1.0, http://mac.mf3x3.com/GIS/GEOTIFF/geotiff_spec.pdf. (accessed 01.05.2018).

Rossetto, R., De Filippis, G., Borsi, I., et al. (2018), Integrating free and open source tools and distributed modelling codes in GIS environment for data-based groundwater management. *Environmental Modelling & Software*, 107, 210-230.

Tian, Y., Zheng, Y., Han, F., et al. (2018), A comprehensive graphical modeling platform designed for integrated hydrological simulation. *Environmental Modelling & Software*, 108, 154-173.

Urbanek, S. (2008), proj4: A Simple Interface to the PROJ. 4 Cartographic Projections Library (R package version 1.0-4).

Voinov, A., Cerco, C. (2010), Model integration and the role of data. *Environmental Modelling & Software*, 25, 965-969.

Wen, Y., Chen, M., Lü, G., et al. (2013), Prototyping an open environment for sharing geographical analysis models on cloud computing platform. *International Journal of Digital Earth*, 6, 356-382.

Wen, Y., Chen, M., Yue, S., et al. (2017), A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment. *International Journal of Digital Earth*, 10, 405-425.

Wen, Y., Lü, G., Yang, H., Cao, D., Chen, M. (2006), Service oriented distributed geological model integrated framework. *Journal of Remote Sensing*, 2, 160-168.

Yang, C., Goodchild, M., Huang, Q., et al. (2011), Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?. *International Journal of Digital Earth*, 4(4), 305-329.

Yin, L., Zhu, J., Zhang, X., et al. (2015), Visual analysis and simulation of dam-break flood spatiotemporal process in a network environment. Environmental Earth Sciences, 74(10), 7133-7146.

Yue, P., Guo, X., Zhang, M., et al. (2016), Linked Data and SDI: The case on Web geoprocessing workflows. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114, 245-257.

Yue, S., Chen, M., Wen, Y., et al. (2016), 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*, 114, 258-273.

Yue, S., Wen, Y., Chen, M., et al. (2015), A data description model for reusing, sharing and integrating geo-analysis models. *Environmental Earth Sciences*, 74, 7081-7099.

Zhao, P., Foerster, T., Yue, P., (2012), The geoprocessing web. *Computers & Geosciences*, 47, 3-12.

Zheng, S. (2016), Research on Water Quality Assimilation Simulation of Taihu Based on Particle Filter. Nanjing Normal University.

Zhu, Y., Zhu, A., Feng, M., et al. (2017), A similarity-based automatic data recommendation approach for geographic models. *International Journal of Geographical Information Science*, 31, 1403-1424.

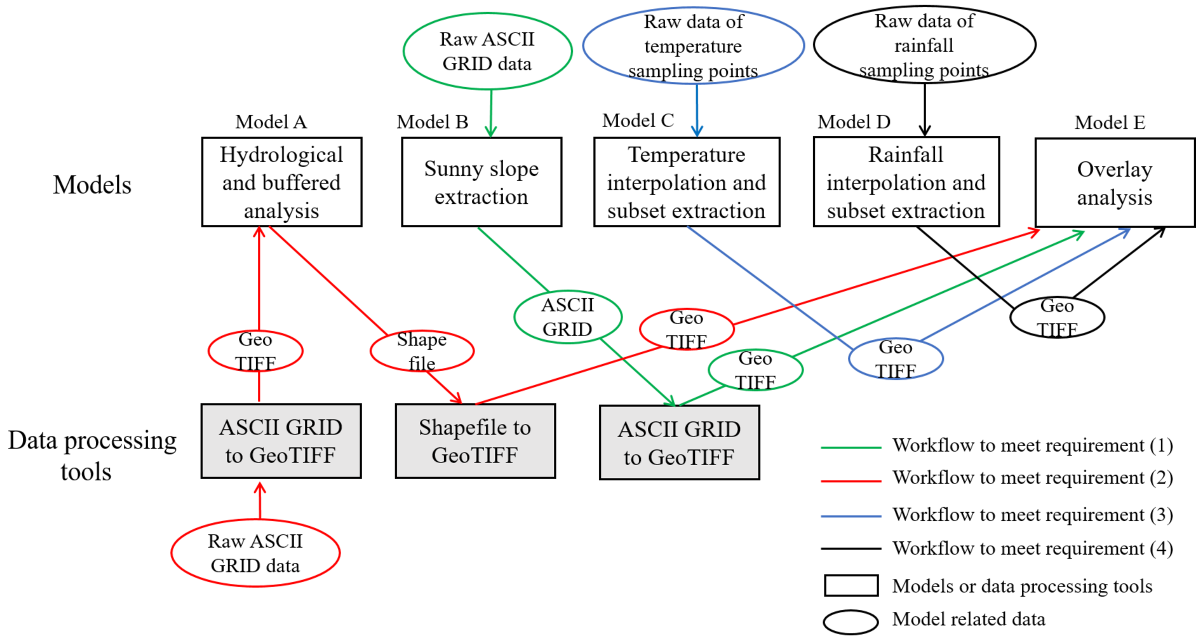
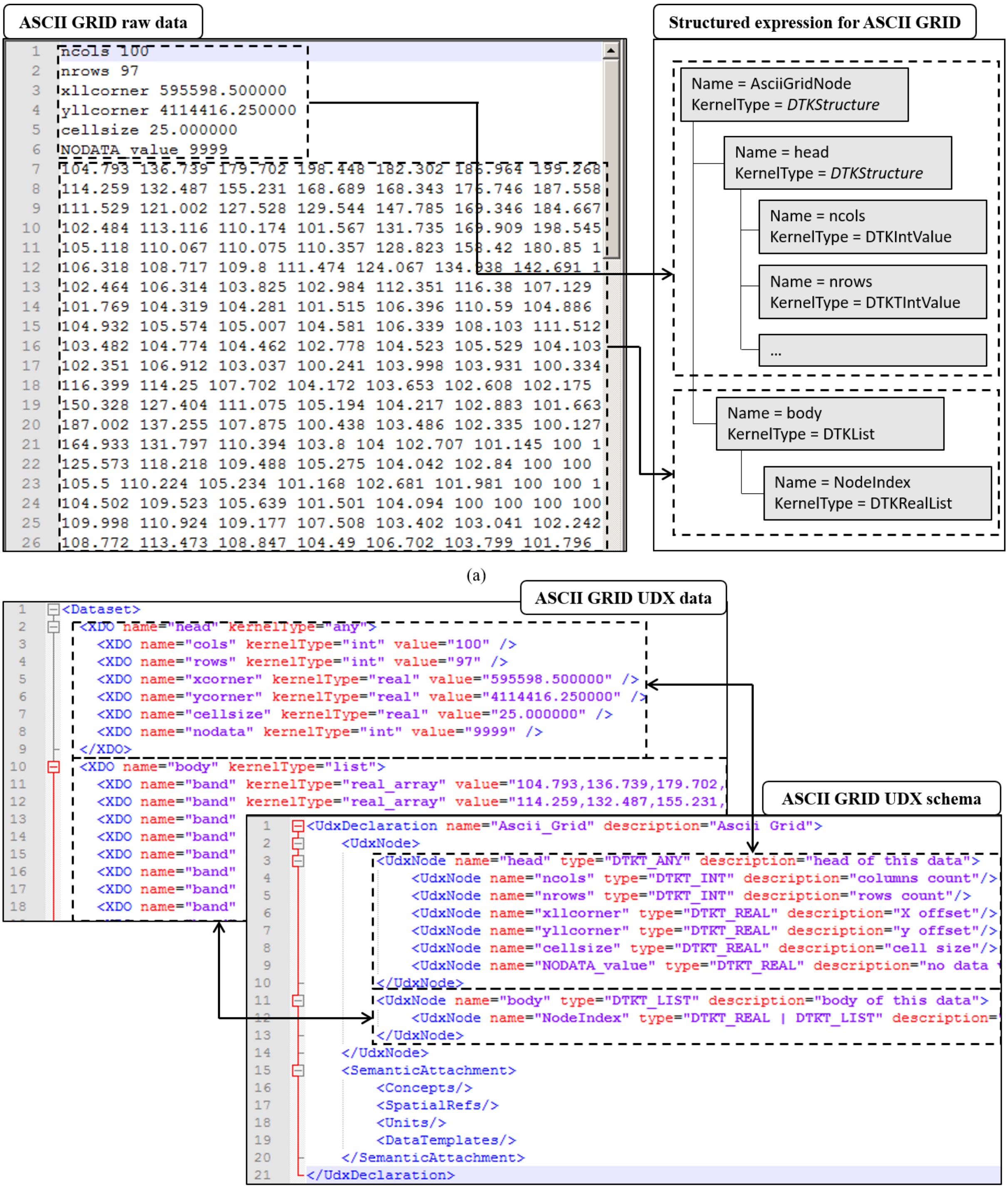

Raw ASCII GRID data
Raw data of temperature sampling points
Raw data of rainfall sampling points
Model A
Model B
Model C
Model D
Model E
Models
Hydrological and buffered analysis
Sunny slope extraction
Temperature interpolation and subset extraction
Rainfall interpolation and subset extraction
Overlay analysis
ASCII GRID
Geo TIFF
Shape file
Geo TIFF
Geo TIFF
Geo TIFF
Geo TIFF
Data processing tools
ASCII GRID to GeoTIFF
Shapefile to GeoTIFF
ASCII GRID to GeoTIFF
Raw ASCII GRID data
Workflow to meet requirement (1)
Workflow to meet requirement (2)
Workflow to meet requirement (3)
Workflow to meet requirement (4)
Models or data processing tools
Model related data

Figure 1. Process of area selection for planting apple trees.



```
ncols 100
nrows 97
xllcorner 595598.500000
yllcorner 4114416.250000
cellsize 25.000000
NODATA_value 9999
104.793 136.739 179.702 198.448 182.302 186.964 199.268
114.259 132.487 155.231 168.689 168.343 176.746 187.558
111.529 121.002 127.528 129.544 147.785 169.346 184.667
102.484 113.116 110.174 101.567 131.735 169.909 198.545
105.118 110.067 110.075 110.357 128.823 158.42 180.85 1
106.318 108.717 109.8 111.474 124.067 134.938 142.691 1
102.464 106.314 103.825 102.984 112.351 116.38 107.129
101.769 104.319 104.281 101.515 106.396 110.59 104.886
104.932 105.574 105.007 104.581 106.339 108.103 111.512
103.482 104.774 104.462 102.778 104.523 105.529 104.103
102.351 106.912 103.037 100.241 103.998 103.931 100.334
116.399 114.25 107.702 104.172 103.653 102.608 102.175
150.328 127.404 111.075 105.194 104.217 102.883 101.663
187.002 137.255 107.875 100.438 103.486 102.335 100.127
164.933 131.797 110.394 103.8 104 102.707 101.145 100 1
125.573 118.218 109.488 105.275 104.042 102.84 100 100
105.5 110.224 105.234 101.168 102.681 101.981 100 100 1
104.502 109.523 105.639 101.501 104.094 100 100 100 100
109.998 110.924 109.177 107.508 103.402 103.041 102.242
108.772 113.473 108.847 104.49 106.702 103.799 101.796
```

(a)

```
<Dataset>
  <XDO name="head" kernelType="any">
    <XDO name="cols" kernelType="int" value="100" />
    <XDO name="rows" kernelType="int" value="97" />
    <XDO name="xcorner" kernelType="real" value="595598.500000" />
    <XDO name="ycorner" kernelType="real" value="4114416.250000" />
    <XDO name="cellsize" kernelType="real" value="25.000000" />
    <XDO name="nodata" kernelType="int" value="9999" />
  </XDO>
  <XDO name="body" kernelType="list">
    <XDO name="band" kernelType="real_array" value="104.793,136.739,179.702,
    <XDO name="band" kernelType="real_array" value="114.259,132.487,155.231,
    <XDO name="band"
    <XDO name="band"
    <XDO name="band"
    <XDO name="band"
    <XDO name="band"
    <XDO name="band"
```

```
<UdxDeclaration name="Ascii_Grid" description="Ascii Grid">
    <UdxNode>
        <UdxNode name="head" type="DTKT_ANY" description="head of this data">
            <UdxNode name="ncols" type="DTKT_INT" description="columns count"/>
            <UdxNode name="nrows" type="DTKT_INT" description="rows count"/>
            <UdxNode name="xllcorner" type="DTKT_REAL" description="X offset"/>
            <UdxNode name="yllcorner" type="DTKT_REAL" description="y offset"/>
            <UdxNode name="cellsize" type="DTKT_REAL" description="cell size"/>
            <UdxNode name="NODATA_value" type="DTKT_REAL" description="no data
        </UdxNode>
        <UdxNode name="body" type="DTKT_LIST" description="body of this data">
            <UdxNode name="NodeIndex" type="DTKT_REAL | DTKT_LIST" description=
        </UdxNode>
    </UdxNode>
    <SemanticAttachment>
        <Concepts/>
        <SpatialRefs/>
        <Units/>
        <DataTemplates/>
    </SemanticAttachment>
</UdxDeclaration>
```

(b)

Figure 2. UDX expression of the ASCII GRID raw data.

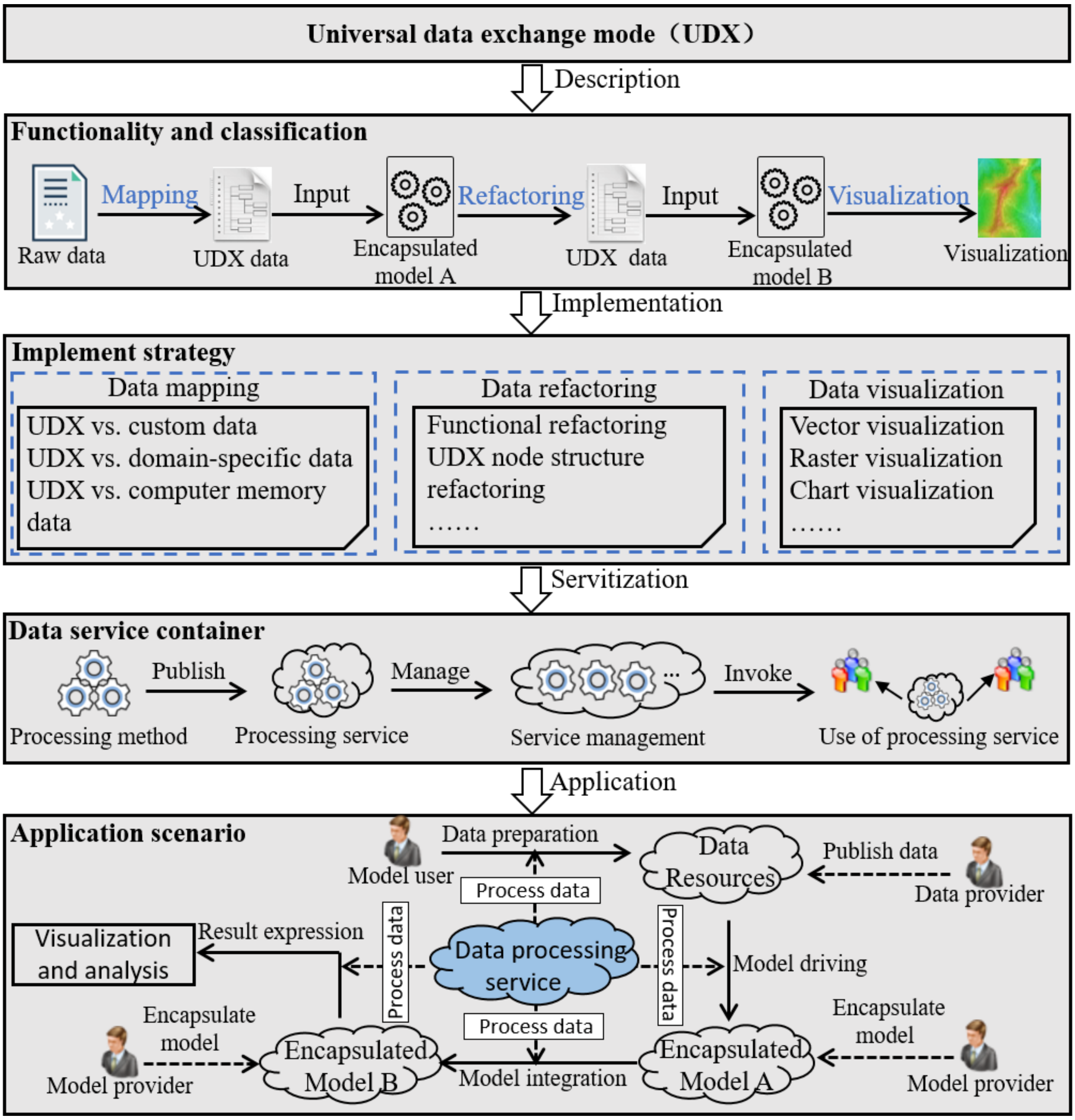


Figure 3. Workflow of data processing services.

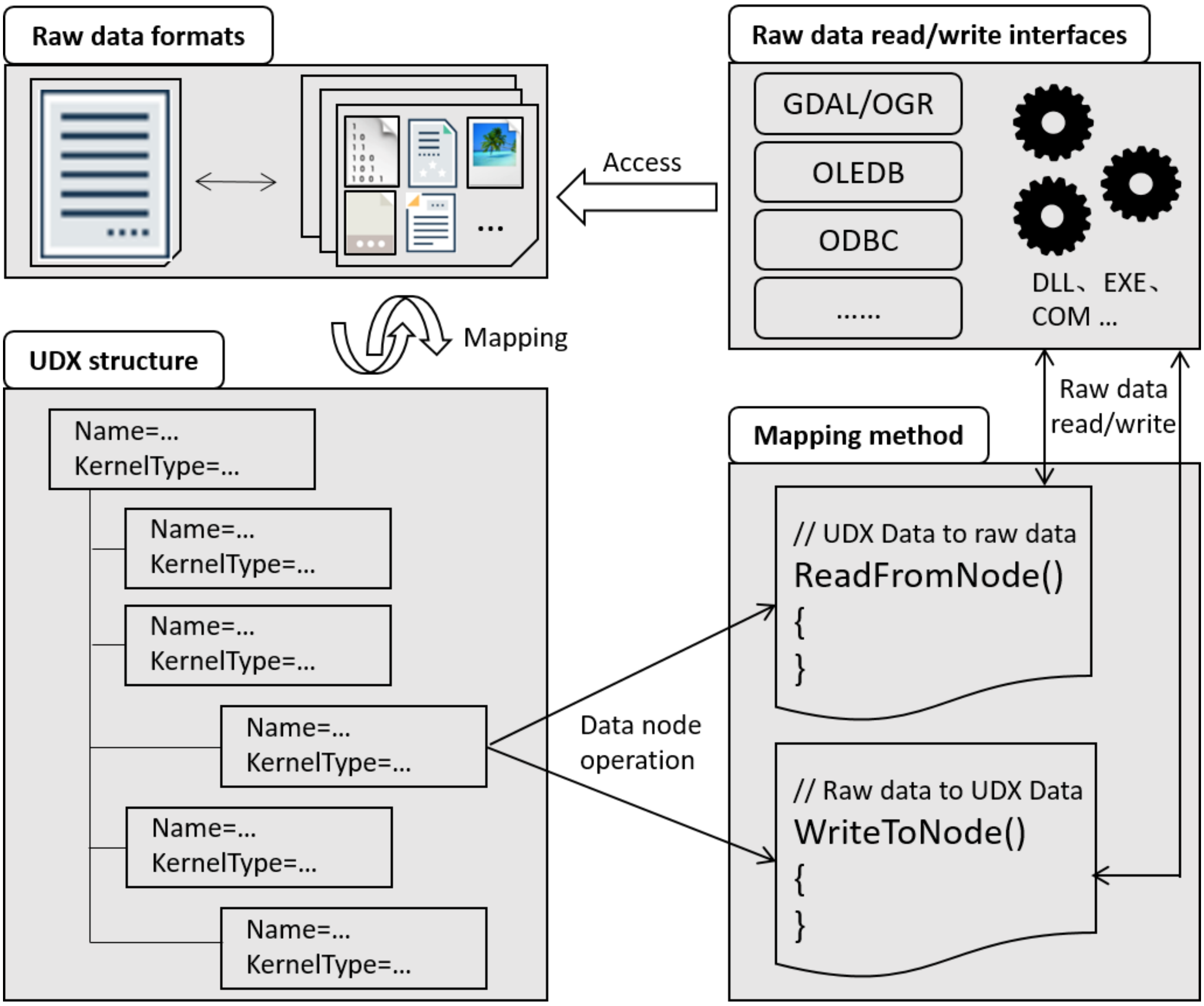
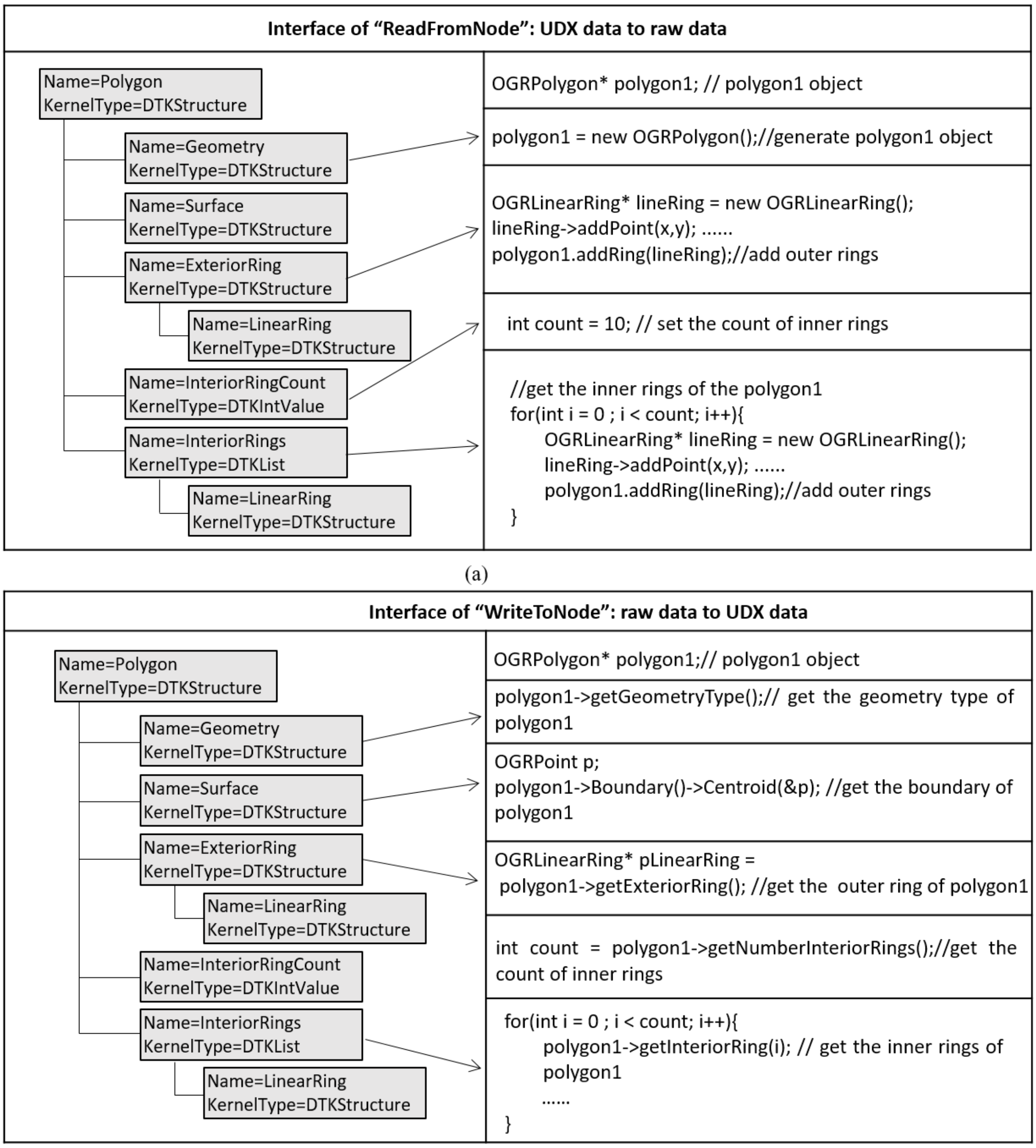


Figure 4. Implementation of the data mapping method.


Interface of "ReadFromNode": UDX data to raw data
Name=Polygon KernelType=DTKStructure
Name=Geometry KernelType=DTKStructure
Name=Surface KernelType=DTKStructure
Name=ExteriorRing KernelType=DTKStructure
Name=LinearRing KernelType=DTKStructure
Name=InteriorRingCount KernelType=DTKIntValue
Name=InteriorRings KernelType=DTKList
Name=LinearRing KernelType=DTKStructure

```
OGRPolygon* polygon1; // polygon1 object

polygon1 = new OGRPolygon();//generate polygon1 object

OGRLinearRing* lineRing = new OGRLinearRing();
lineRing->addPoint(x,y); ......
polygon1.addRing(lineRing);//add outer rings

  int count = 10; // set the count of inner rings

  //get the inner rings of the polygon1
  for(int i = 0 ; i < count; i++){
       OGRLinearRing* lineRing = new OGRLinearRing();
       lineRing->addPoint(x,y); ......
       polygon1.addRing(lineRing);//add outer rings
  }
```

(a)

Interface of "WriteToNode": raw data to UDX data
Name=Polygon KernelType=DTKStructure
Name=Geometry KernelType=DTKStructure
Name=Surface KernelType=DTKStructure
Name=ExteriorRing KernelType=DTKStructure
Name=LinearRing KernelType=DTKStructure
Name=InteriorRingCount KernelType=DTKIntValue
Name=InteriorRings KernelType=DTKList
Name=LinearRing KernelType=DTKStructure

```
OGRPolygon* polygon1;// polygon1 object

polygon1->getGeometryType();// get the geometry type of
polygon1

OGRPoint p;
polygon1->Boundary()->Centroid(&p); //get the boundary of
polygon1

OGRLinearRing* pLinearRing =
 polygon1->getExteriorRing(); //get the  outer ring of polygon1

int count = polygon1->getNumberInteriorRings();//get the
count of inner rings

 for(int i = 0 ; i < count; i++){
      polygon1->getInteriorRing(i); // get the inner rings of
      polygon1
      ......
 }
```

(b)

Figure 5. Pseudocode of the data mapping method for shapefiles.

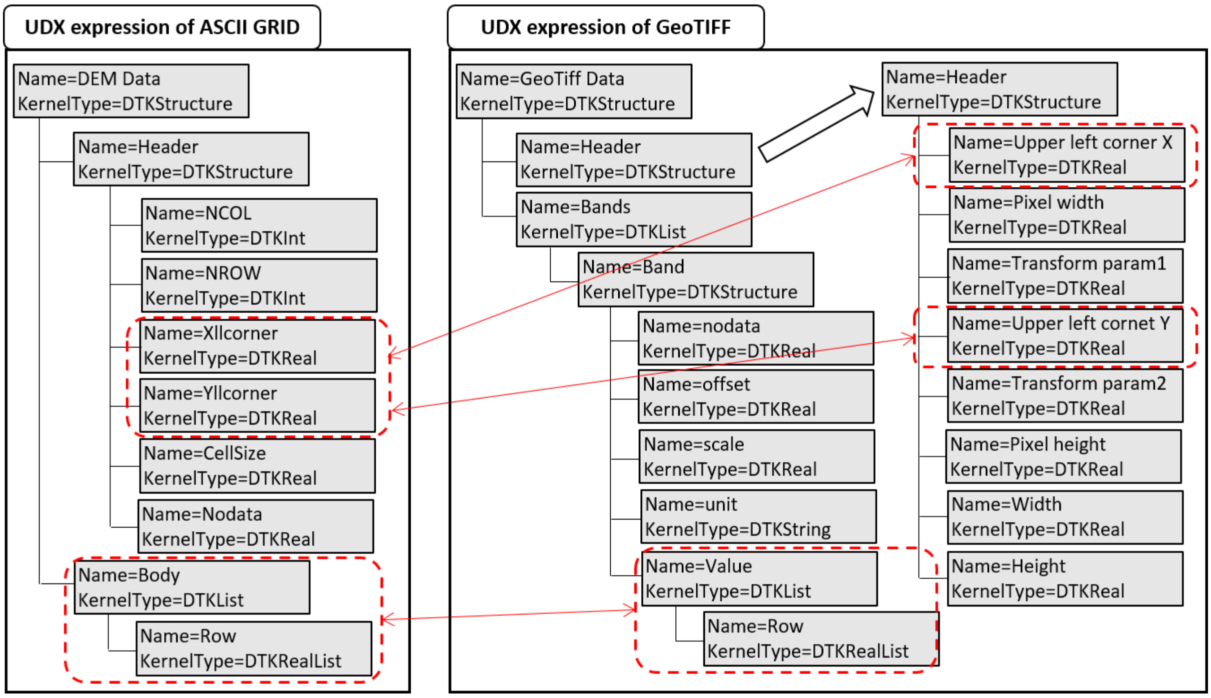

UDX expression of ASCII GRID
Name=DEM Data KernelType=DTKStructure
Name=Header KernelType=DTKStructure
Name=NCOL KernelType=DTKInt
Name=NROW KernelType=DTKInt
Name=Xllcorner KernelType=DTKReal
Name=Yllcorner KernelType=DTKReal
Name=CellSize KernelType=DTKReal
Name=Nodata KernelType=DTKReal
Name=Body KernelType=DTKList
Name=Row KernelType=DTKRealList
UDX expression of GeoTIFF
Name=GeoTiff Data KernelType=DTKStructure
Name=Header KernelType=DTKStructure
Name=Bands KernelType=DTKList
Name=Band KernelType=DTKStructure
Name=nodata KernelType=DTKReal
Name=offset KernelType=DTKReal
Name=scale KernelType=DTKReal
Name=unit KernelType=DTKString
Name=Value KernelType=DTKList
Name=Row KernelType=DTKRealList
Name=Header KernelType=DTKStructure
Name=Upper left corner X KernelType=DTKReal
Name=Pixel width KernelType=DTKReal
Name=Transform param1 KernelType=DTKReal
Name=Upper left cornet Y KernelType=DTKReal
Name=Transform param2 KernelType=DTKReal
Name=Pixel height KernelType=DTKReal
Name=Width KernelType=DTKReal
Name=Height KernelType=DTKReal

Figure 6. Refactoring of data nodes between the ASCII GRID and GeoTIFF formats.

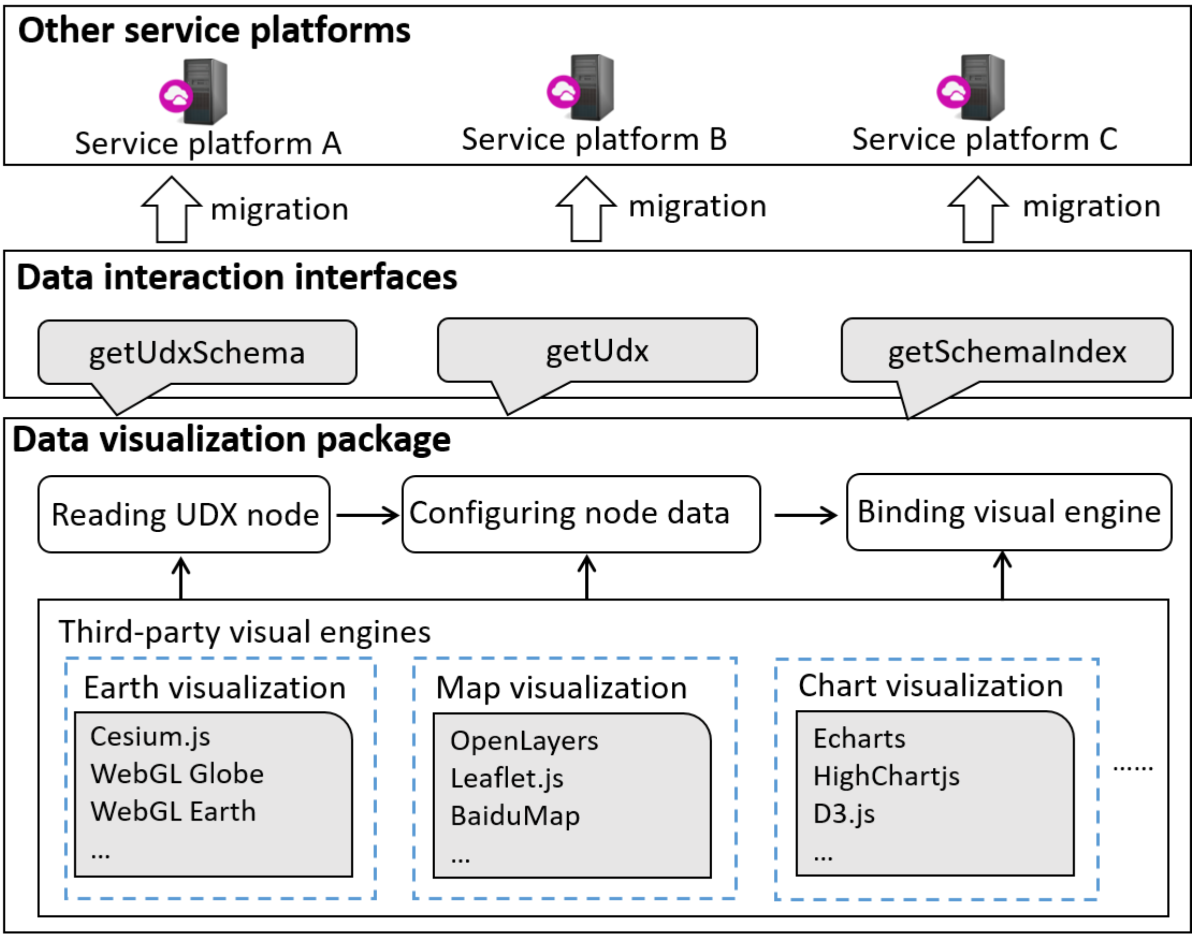


Figure 7. Implementation of the visual package.

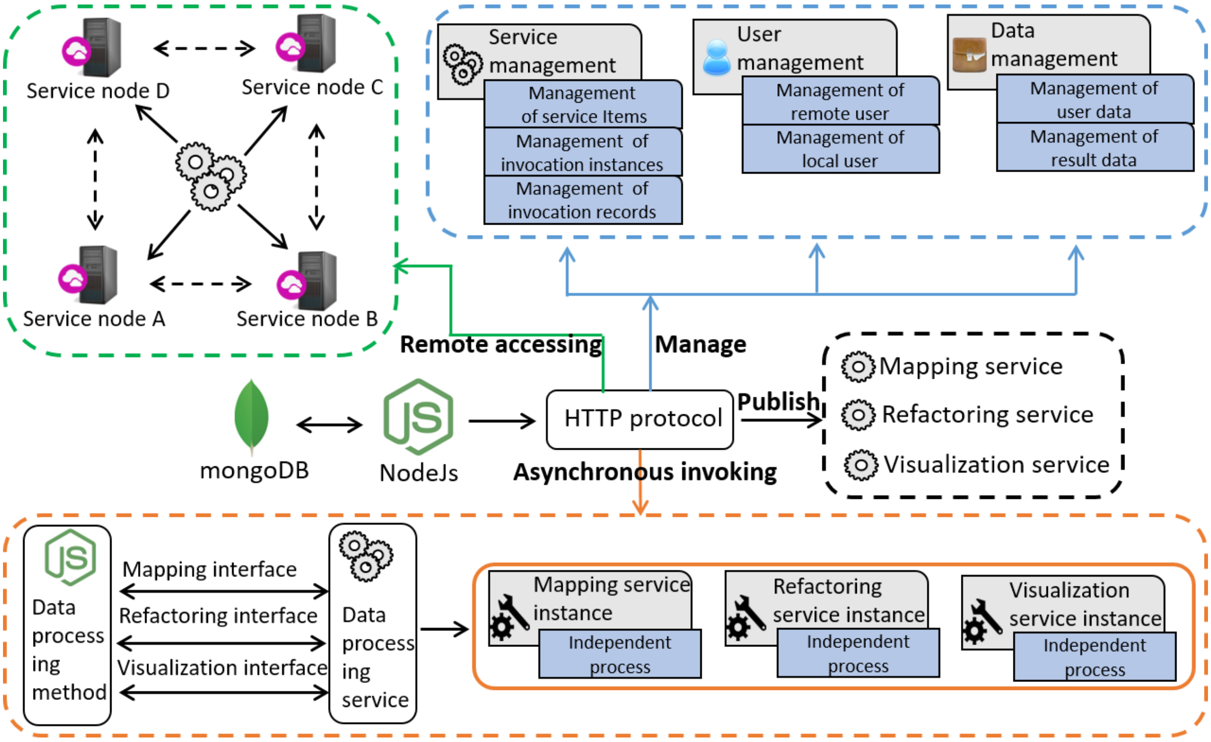
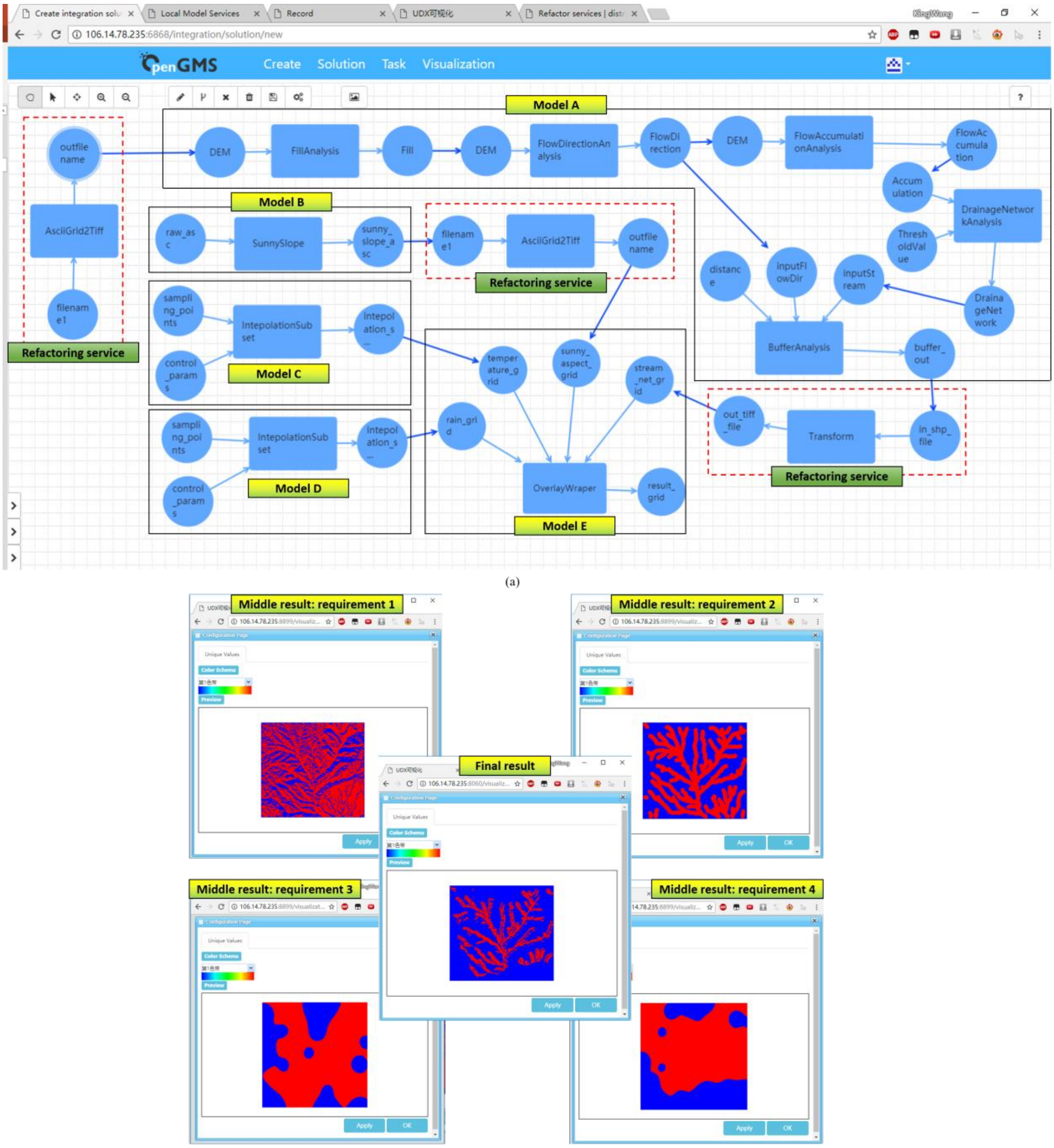


Figure 8. Architecture of the data service container.



(a)

(b)

Figure 9. Implementation of the model integration scenario.

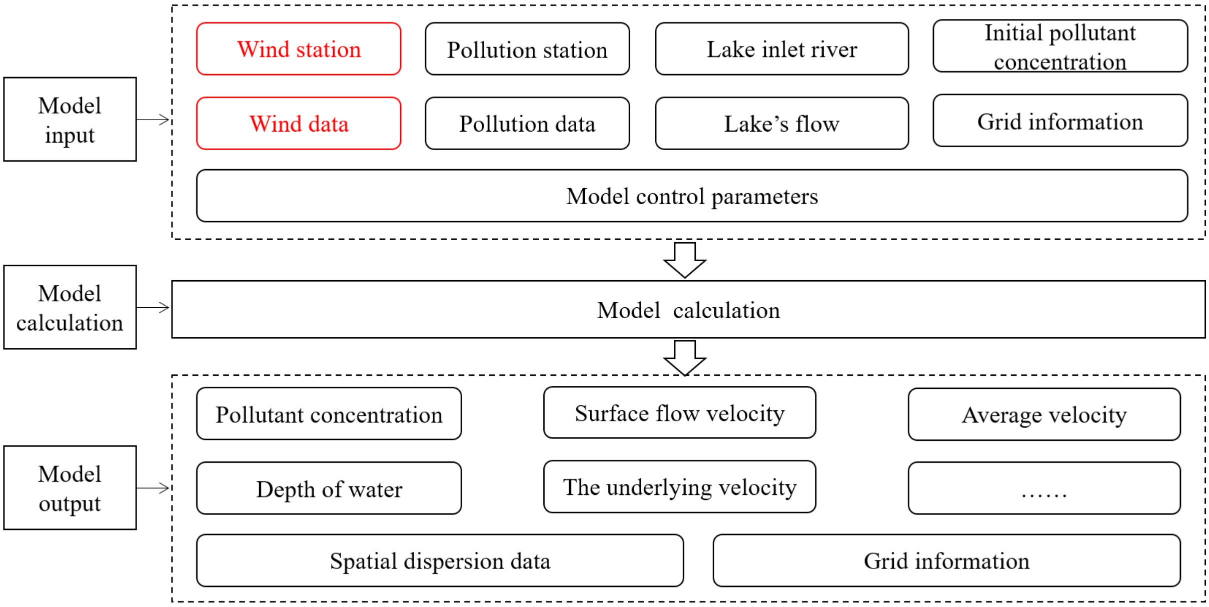
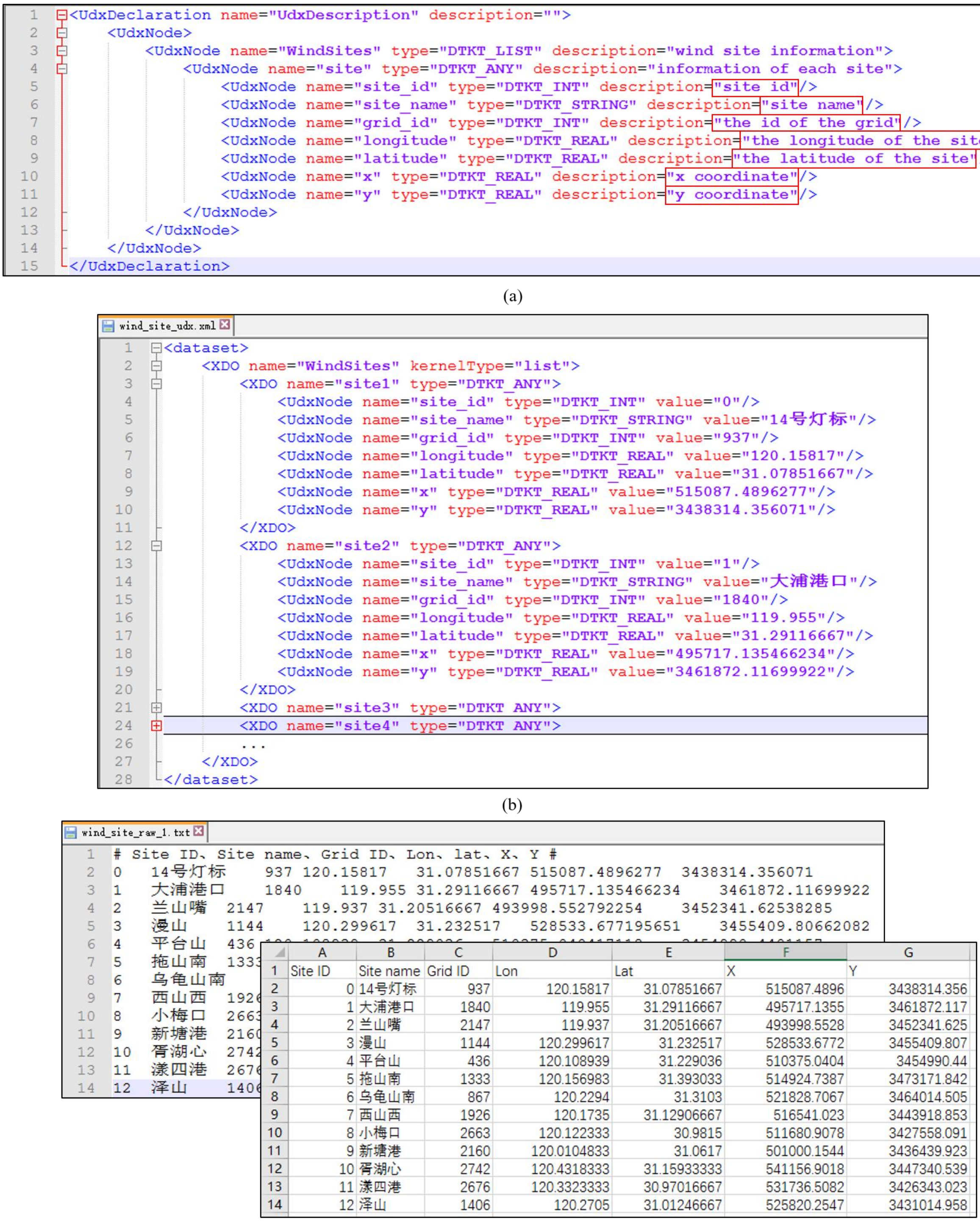


Figure 10. Framework of the TWPDM.



```
<UdxDeclaration name="UdxDescription" description="">
    <UdxNode>
        <UdxNode name="WindSites" type="DTKT_LIST" description="wind site information">
            <UdxNode name="site" type="DTKT_ANY" description="information of each site">
                <UdxNode name="site_id" type="DTKT_INT" description="site id"/>
                <UdxNode name="site_name" type="DTKT_STRING" description="site name"/>
                <UdxNode name="grid_id" type="DTKT_INT" description="the id of the grid"/>
                <UdxNode name="longitude" type="DTKT_REAL" description="the longitude of the site"/>
                <UdxNode name="latitude" type="DTKT_REAL" description="the latitude of the site"/>
                <UdxNode name="x" type="DTKT_REAL" description="x coordinate"/>
                <UdxNode name="y" type="DTKT_REAL" description="y coordinate"/>
            </UdxNode>
        </UdxNode>
    </UdxNode>
</UdxDeclaration>
```

(a)

```
<dataset>
    <XDO name="WindSites" kernelType="list">
        <XDO name="site1" type="DTKT_ANY">
            <UdxNode name="site_id" type="DTKT_INT" value="0"/>
            <UdxNode name="site_name" type="DTKT_STRING" value="14号灯标"/>
            <UdxNode name="grid_id" type="DTKT_INT" value="937"/>
            <UdxNode name="longitude" type="DTKT_REAL" value="120.15817"/>
            <UdxNode name="latitude" type="DTKT_REAL" value="31.07851667"/>
            <UdxNode name="x" type="DTKT_REAL" value="515087.4896277"/>
            <UdxNode name="y" type="DTKT_REAL" value="3438314.356071"/>
        </XDO>
        <XDO name="site2" type="DTKT_ANY">
            <UdxNode name="site_id" type="DTKT_INT" value="1"/>
            <UdxNode name="site_name" type="DTKT_STRING" value="大浦港口"/>
            <UdxNode name="grid_id" type="DTKT_INT" value="1840"/>
            <UdxNode name="longitude" type="DTKT_REAL" value="119.955"/>
            <UdxNode name="latitude" type="DTKT_REAL" value="31.29116667"/>
            <UdxNode name="x" type="DTKT_REAL" value="495717.135466234"/>
            <UdxNode name="y" type="DTKT_REAL" value="3461872.11699922"/>
        </XDO>
        <XDO name="site3" type="DTKT_ANY">
        <XDO name="site4" type="DTKT_ANY">
        ...
    </XDO>
</dataset>
```

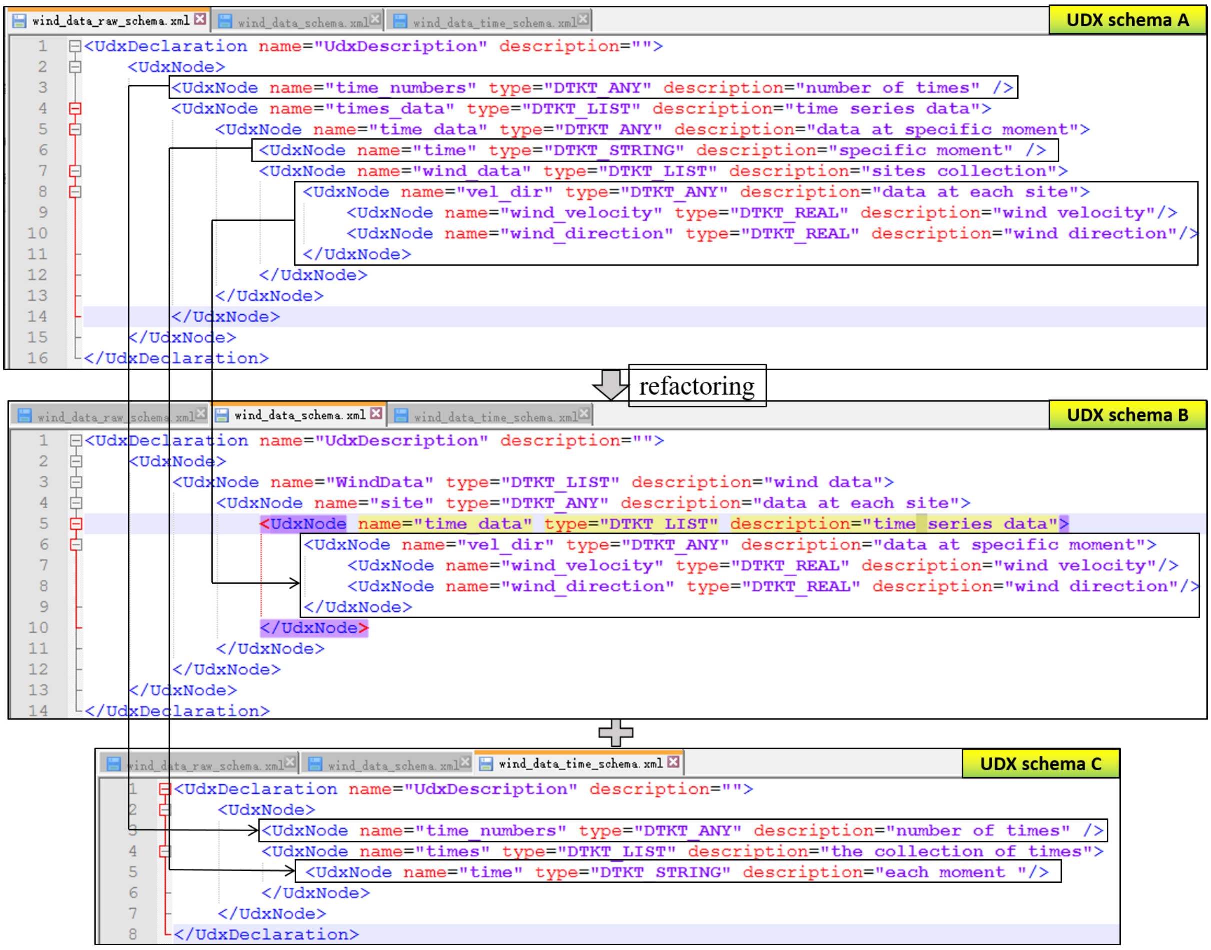
(b)

```
# Site ID、Site name、Grid ID、Lon、lat、X、Y #
0   14号灯标    937 120.15817   31.07851667 515087.4896277  3438314.356071
1   大浦港口    1840    119.955 31.29116667 495717.135466234    3461872.11699922
2   兰山嘴  2147    119.937 31.20516667 493998.552792254    3452341.62538285
3   漫山    1144    120.299617  31.232517   528533.677195651    3455409.80662082
4   平台山  436
5   拖山南  1333
6   乌龟山南
7   西山西  1926
8   小梅口  2663
9   新塘港  2160
10  胥湖心  2742
11  漾四港  2676
12  泽山    1406
```

	A	B	C	D	E	F	G
1	Site ID	Site name	Grid ID	Lon	Lat	X	Y
2	0	14号灯标	937	120.15817	31.07851667	515087.4896	3438314.356
3	1	大浦港口	1840	119.955	31.29116667	495717.1355	3461872.117
4	2	兰山嘴	2147	119.937	31.20516667	493998.5528	3452341.625
5	3	漫山	1144	120.299617	31.232517	528533.6772	3455409.807
6	4	平台山	436	120.108939	31.229036	510375.0404	3454990.44
7	5	拖山南	1333	120.156983	31.393033	514924.7387	3473171.842
8	6	乌龟山南	867	120.2294	31.3103	521828.7067	3464014.505
9	7	西山西	1926	120.1735	31.12906667	516541.023	3443918.853
10	8	小梅口	2663	120.122333	30.9815	511680.9078	3427558.091
11	9	新塘港	2160	120.0104833	31.0617	501000.1544	3436439.923
12	10	胥湖心	2742	120.4318333	31.15933333	541156.9018	3447340.539
13	11	漾四港	2676	120.3323333	30.97016667	531736.5082	3426343.023
14	12	泽山	1406	120.2705	31.01246667	525820.2547	3431014.958

(c)

Figure 11. Preparation of the wind-site data.

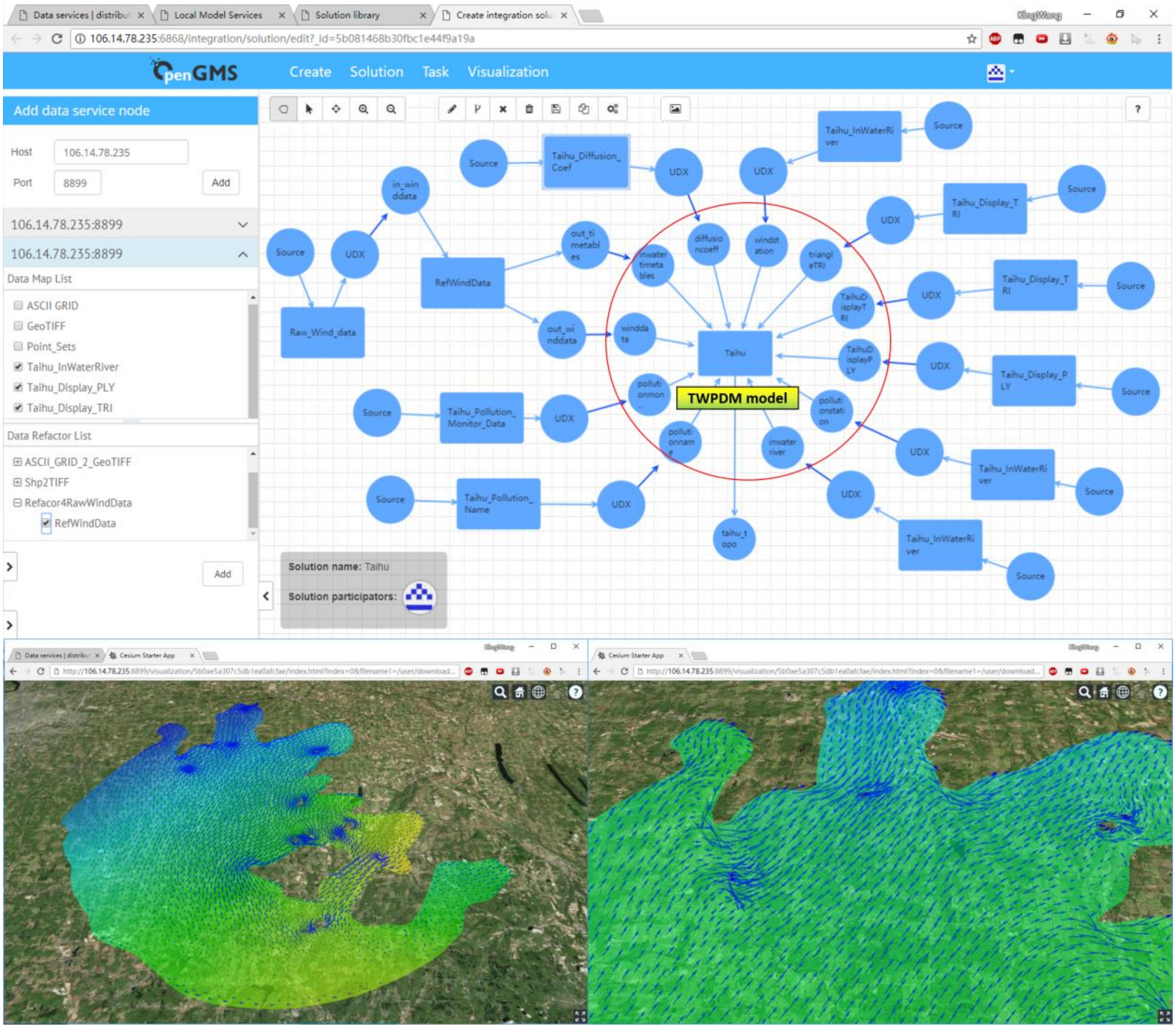


```
<UdxDeclaration name="UdxDescription" description="">
    <UdxNode>
        <UdxNode name="time numbers" type="DTKT ANY" description="number of times" />
        <UdxNode name="times_data" type="DTKT_LIST" description="time series data">
            <UdxNode name="time data" type="DTKT ANY" description="data at specific moment">
                <UdxNode name="time" type="DTKT_STRING" description="specific moment" />
                <UdxNode name="wind data" type="DTKT LIST" description="sites collection">
                    <UdxNode name="vel_dir" type="DTKT_ANY" description="data at each site">
                        <UdxNode name="wind_velocity" type="DTKT_REAL" description="wind velocity"/>
                        <UdxNode name="wind_direction" type="DTKT_REAL" description="wind direction"/>
                    </UdxNode>
                </UdxNode>
            </UdxNode>
        </UdxNode>
    </UdxNode>
</UdxDeclaration>
```

```
<UdxDeclaration name="UdxDescription" description="">
    <UdxNode>
        <UdxNode name="WindData" type="DTKT_LIST" description="wind data">
            <UdxNode name="site" type="DTKT_ANY" description="data at each site">
                <UdxNode name="time data" type="DTKT LIST" description="time series data">
                    <UdxNode name="vel_dir" type="DTKT_ANY" description="data at specific moment">
                        <UdxNode name="wind_velocity" type="DTKT_REAL" description="wind velocity"/>
                        <UdxNode name="wind_direction" type="DTKT_REAL" description="wind direction"/>
                    </UdxNode>
                </UdxNode>
            </UdxNode>
        </UdxNode>
    </UdxNode>
</UdxDeclaration>
```

```
<UdxDeclaration name="UdxDescription" description="">
    <UdxNode>
        <UdxNode name="time numbers" type="DTKT ANY" description="number of times" />
        <UdxNode name="times" type="DTKT_LIST" description="the collection of times">
            <UdxNode name="time" type="DTKT STRING" description="each moment "/>
        </UdxNode>
    </UdxNode>
</UdxDeclaration>
```

Figure 12. Data refactoring when preparing the wind data.



Host 106.14.78.235

Port 8899

Solution name: Taihu

Solution participators:

Figure 13. Model invocation and result visualization.