Integrating three lake models into a Phytoplankton Prediction System for Lake Taihu (Taihu PPS) with Python

Jiacong Huang, Junfeng Gao, Georg Hörmann and Wolf M. Mooij

ABSTRACT

In the past decade, much work has been done on integrating different lake models using general frameworks to overcome model incompatibilities. However, a framework may not be flexible enough to support applications in different fields. To overcome this problem, we used Python to integrate three lake models into a Phytoplankton Prediction System for Lake Taihu (Taihu PPS). The system predicts the short-term (1–4 days) distribution of phytoplankton biomass in this large eutrophic lake in China. The object-oriented scripting language Python is used as the so-called 'glue language' (a programming language used for connecting software components). The distinguishing features of Python include rich extension libraries for spatial and temporal modelling, modular software architecture, free licensing and a high performance resulting in short execution time. These features facilitate efficient integration of the three models into Taihu PPS. Advanced tools (e.g. tools for statistics, 3D visualization and model calibration) could be developed in the future with the aid of the continuously updated Python libraries. Taihu PPS simulated phytoplankton biomass well and has already been applied to support decision making.

Key words | Lake Taihu, model integration, Phytoplankton Prediction System, Python, Python library

Jiacong Huang
Junfeng Gao (corresponding author)
Nanjing Institute of Geography and Limnology,
Chinese Academy of Sciences,
73 East Beijing Road, Nanjing 210008, China
E-mail: *gaojunf@niglas.ac.cn*

Jiacong Huang
Graduate University of the Chinese Academy of Sciences, Beijing 100049, China

Jiacong Huang
Georg Hörmann
Department of Hydrology and Water Resources Management,
Institute of Natural Resources Conservation,
Kiel University, Kiel 24118, Germany

Wolf M. Mooij
Netherlands Institute of Ecology (NIOO-KNAW),
Department of Aquatic Ecology, P.O. Box 50,
6700 AB Wageningen, The Netherlands
and
Wageningen University,
Department of Aquatic Ecology and Water Quality Management, P.O. Box 47,
6700 AA Wageningen, The Netherlands

INTRODUCTION

The integration of model components that stem from different scientific disciplines is important for making optimal use of the existing knowledge on the functioning of complex ecosystems (Argent 2004). However, existing models are sometimes incompatible with each other and are difficult to couple (Holzworth *et al*. 2010). The compatibility problem is mostly due to different spatial and temporal scales, modelling techniques, programming languages, software platforms and data types used in various model components (Argent 2004; Oxley *et al*. 2004; Roberts *et al*. 2010). Therefore, an approach that is capable of efficiently coupling model components is much needed.

Models can be integrated using programming languages (the language approach) or frameworks (the framework approach). Some intermediate approaches have been proposed by using frameworks within a programming language (e.g. Karssenberg *et al*. 2010). Among these approaches, the framework approach is the most widely used (van Kouwen *et al*. 2008; Govind *et al*. 2009; Schmitz *et al*. 2009). Some component-based modelling frameworks, such as OpenMI (Gregersen *et al*. 2007), ICMS, Tarsier (Rahman 2004; Argent *et al*. 2006) and APSIM (Holzworth *et al*. 2010), have been used to integrate many modules and have been applied in different fields. Other frameworks, such as MGET, are dependent on other software and save development time at the cost of additional compatibility problems (Roberts *et al*. 2010).

However, a framework is typically specific to a given field, and may not be flexible enough to support applications in other fields (Karssenberg 2002). Researchers in many cases need features not available in the framework, so they have to develop the required functions themselves.

doi: 10.2166/hydro.2011.020

Moreover, the framework is normally designed by software engineers, which makes it difficult for modellers to make modifications. In addition, the cost of acquiring a licence of the framework may also limit its use.

As an alternative to the framework approach, we used a scripting language (Python) for model integration. As scripting languages are designed for connecting software components, an application can be developed five to ten times faster than with a traditional system programming language (Ousterhout 1998). Python supports a diverse collection of free libraries and multiple programming languages, and is independent of other software (Karssenberg *et al.* 2007). These features allow Python to solve problems in integrating model components from different fields. Despite its obvious benefits, however, Python has been scarcely used for coupling models (Kraft *et al.* 2010).

The objectives of this paper are to integrate three lake models with Python, and to develop a Phytoplankton Prediction System for Lake Taihu (Taihu PPS) for forecasting the phytoplankton distribution over a time horizon of a few days. Such forecasting helps managers to take emergency measures in time to protect sensitive areas (e.g. intake points of drinking water) and thereby avoid or minimize the negative impact of harmful algal blooms. Taihu PPS has already been used by managers of Lake Taihu in recent years. The following section describes the model components of Taihu PPS. The third section of the paper presents details of Taihu PPS, including the architecture, the 'glue language', used Python libraries and modelling tools and finally the workflow of Taihu PPS. In the fourth section, the paper evaluates the performance of Python in model integration. Finally, conclusions are drawn in the last section.

MODEL COMPONENTS

Models are more than just software components and require effort to link them efficiently to data (Voinov & Cerco 2010). They can be developed from scratch or from existing models (for an overview of many lake models see Mooij *et al.* 2010). Model components developed from scratch have the benefit of full control of the model design and linkage, but require large investments in development and testing. Modelling based on existing models saves development time, but requires additional efforts for model coupling (Lam *et al.* 2004). The latter approach was used in this study.

Algal blooms in Lake Taihu, a large eutrophic lake in China (Figure 1), have caused severe problems in the past decade (Guo 2007). To predict phytoplankton biomass, we combined an existing two-dimensional horizontal hydrodynamic model, a phytoplankton kinetics model and a phytoplankton transport model. Phytoplankton biomass was expressed as chlorophyll-α concentration (Riley & Stefan 1988; Hamilton & Schladow 1997).

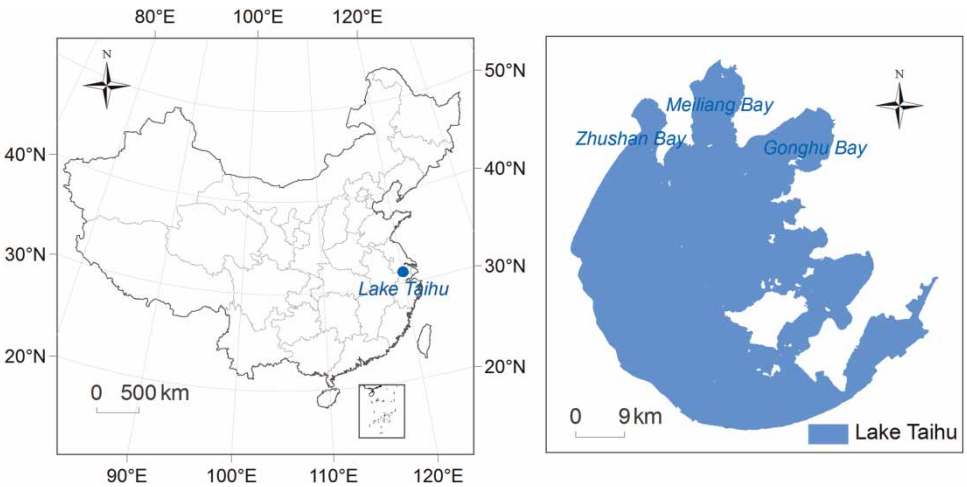

Figure 1 | Location of Lake Taihu in China.

We first identified the linkages and flow of data between the model components (Figure 2). The hydrodynamic model of Lake Taihu (HD Taihu) simulates the water flows in two-dimensional horizontal directions. The output of HD Taihu serves as the input for the phytoplankton transport model to simulate the phytoplankton transport in Lake Taihu. Phytoplankton biomass is calculated with the phytoplankton kinetics model. Interactions and communications among the models are implemented through data import and export. Model inputs and outputs are stored and managed in a database. As this paper focuses on model integration, we here document the programming, interactions and scales (i.e. spatial and temporal scales) of the model components. Details of the model equations, model calibration and model uncertainty can be found in Huang *et al.* (2011).

Hydrodynamic model in Lake Taihu (HD Taihu)

As Lake Taihu is large (surface area 2,338 km^2) and shallow (mean depth 1.9 m) (Qin *et al.* 2007), attention has been paid in previous studies to the spatial heterogeneity of its phytoplankton community. A two-dimensional horizontal hydrodynamic model (HD Taihu) was used to simulate the vertically averaged water velocity in Lake Taihu (Cheng *et al.* 2006). Vertical profiles of the horizontal water velocity were then calculated using an empirical parabola (Cheng *et al.* 2006; Huang *et al.* 2011). As the hydrodynamic computation requires relatively short time steps for numeric stability and accuracy, a time step of 200 s was chosen for HD Taihu. Lake Taihu was divided into 280 longitudinal segments from west to east and 280 latitudinal segments from north to south, giving 36,250 grid cells in the model with a spatial resolution of 250×250 m. A comprehensive description of HD Taihu can be found in Cheng *et al.* (2006).

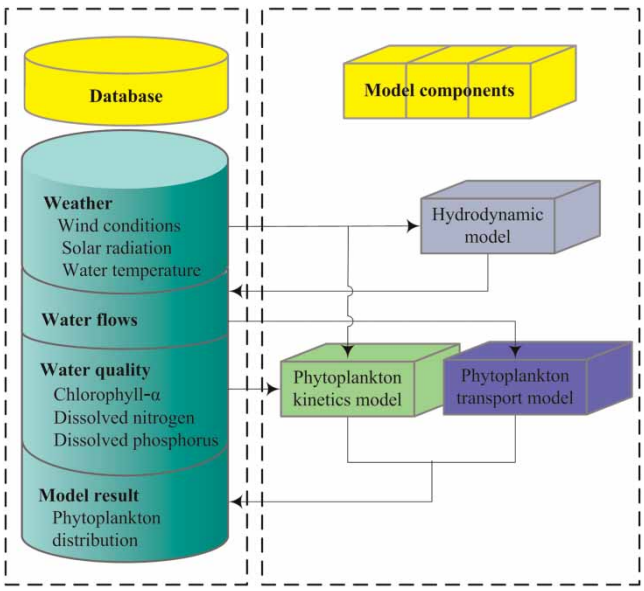

Figure 2 | Schematic diagram of the model component interactions in Taihu PPS.

Phytoplankton kinetics model

The phytoplankton kinetics model describes the biological and physical processes of phytoplankton growth, mortality, respiration, excretion, grazing and sinking. The responses of phytoplankton to photosynthetically active radiation (PAR), water temperature and nutrients (dissolved phosphorus and nitrogen) are covered by the model. PAR was calculated from solar altitude and azimuth and adjusted with cloud cover. Water temperature and nutrients were taken from measured data. A time step of one day and the same spatial resolution as HD Taihu were used in the phytoplankton kinetics model.

Phytoplankton transport model

The phytoplankton transport between different grid cells depends on the vertical profiles of the horizontal water velocity in all grid cells from HD Taihu (referred to as 'water flows' in). Two assumptions were made in this model. The whole phytoplankton biomass is assumed to remain at the water surface when wind velocity is less than 2 m s^{-1}, and when wind velocity is more than 2 m s^{-1}, the wind stress can break up the phytoplankton surface bloom. Subsequently the phytoplankton is treated as if it were dissolved matter that simultaneously follows water movement. These two assumptions were considered to be important for phytoplankton modelling (Hu *et al.* 2006). The transport of phytoplankton is important for the short-term phytoplankton dynamics in Lake Taihu (Wu *et al.* 2010), so a short time step (200 seconds) was used in the phytoplankton transport model.

MODELLING SYSTEM – TAIHU PPS

Architecture of Taihu PPS

The architecture of Taihu PPS consists of an input layer, a modelling engine layer and an output layer (Figure 3). Graphical user interfaces (GUIs) for model input and output were also developed.

The input layer prepares the data for the modelling engine layer. Data sets can be provided in different formats including Microsoft Excel, raster and text. They can be handled by Python extension libraries. The win32com module (also known as pywin32) is used for windows extensions in Python (Hammond & Denn 2000). Geospatial Data Abstraction Library (GDAL) is a translator library for raster geospatial data formats. Text files can be imported directly by Python without an extension library.

The modelling engine layer is the core of Taihu PPS for exchange of data and information. The model components were coupled in this layer with Python. The two-dimensional hydrodynamic model was programmed with FORTRAN (*Dydro.f*), and then compiled into a Python library (*Hydro.pyd*) by F2py (Fortran to Python interface generator). *Hydro.pyd* was then wrapped into a component (hydro) by redesigning its interface. The phytoplankton transport model (PhyTran) and the phytoplankton kinetics model (PhyKinetics) were implemented in Python using GDAL, PCRaster Python and NumPy. In addition to the Python libraries, we developed our own library (*Toolbox.py*) with additional functions needed for

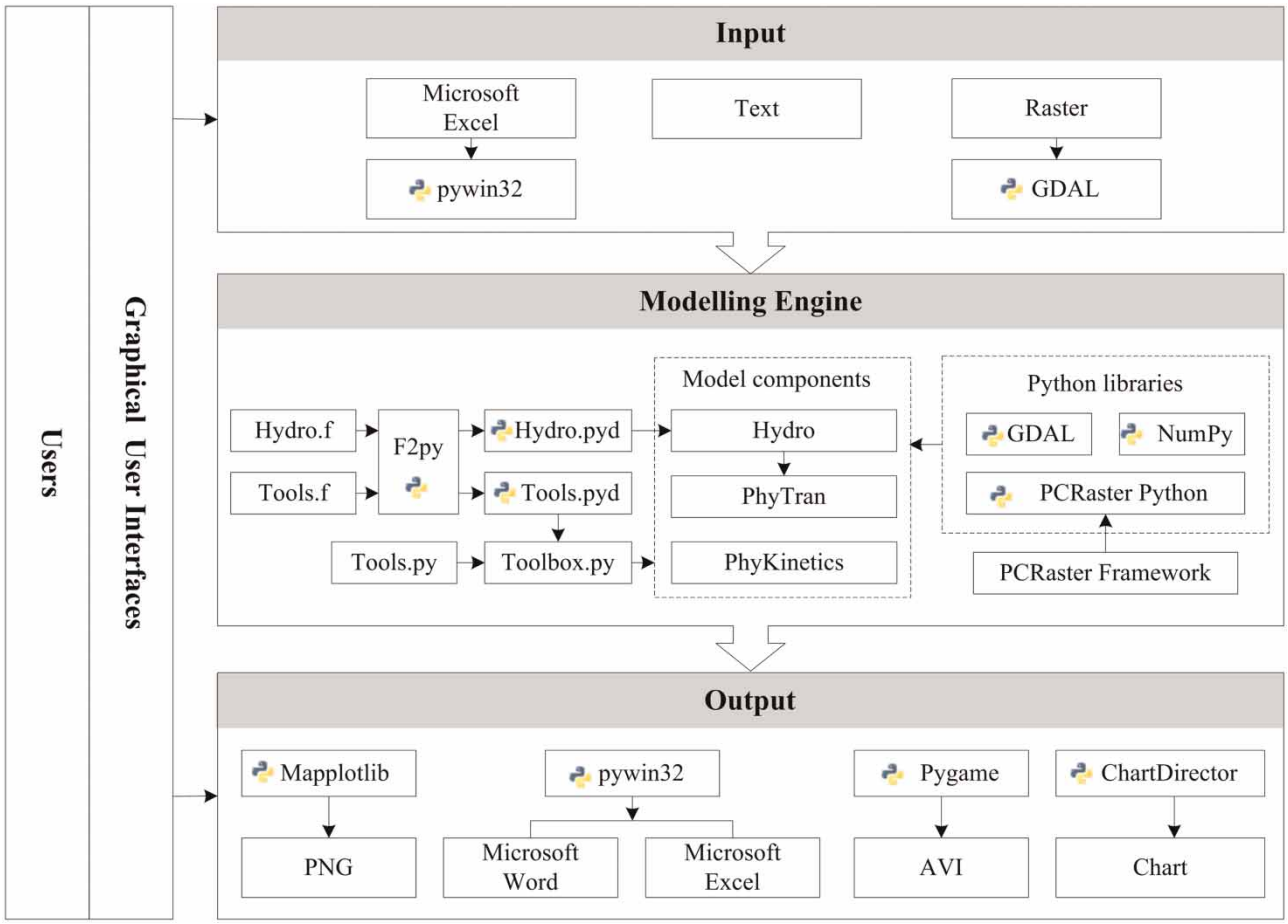

Figure 3 | Software architecture of Taihu PPS showing the input layer, the modelling engine layer, the output layer and interactions between them. The symbol represents the relevant Python library. GDAL: Geospatial Data Abstraction Library; F2py: Fortran to Python interface generator; PNG: Portable Network Graphics; AVI: Audio Video Interleave.

modelling. *Toolbox.py* consists of *Tools.pyd* for math-intensive tasks and *Tools.py* for data manipulation.

The model components were contained in three classes, PhyKinetics, PhyTran and Hydro, with their interfaces shown in Table 1. The interfaces define model inputs and outputs, and hide unnecessary details, such as data conversion and numerical solution of the governing equations in the hydrodynamic component. Because PCRaster Python offers a list of operations that operate on spatial and temporal entities (Schmitz *et al.* 2009), PCRaster maps were mostly used to represent the spatial heterogeneity of the model variables and parameters. This data type is commonly used in spatial modelling, and can also be manipulated by other software (e.g. ArcGIS). The array data type was used in case functionalities needed were unavailable in PCRaster Python but provided in NumPy.

The output layer analyses and displays the output data with Python libraries such as Matplotlib, ChartDirector, pywin32 and Pygame. Taihu PPS is capable of saving the model results as Portable Network Graphics (PNG), Microsoft Word, Microsoft Excel, and Audio Video Interleave (AVI).

Glue language

A system programming language generally runs faster than a scripting language and is better suited to build data structures and algorithms from scratch (Ousterhout 1998; Saenza Jon *et al.* 2002; Galiano *et al.* 2010). In contrast, scripting languages are suitable for model integration. For system integration, applications can be developed five to ten times faster with a scripting language than with a traditional system programming language (Ousterhout 1998). For this reason, scripting languages are sometimes referred to as glue languages or system integration languages (Ousterhout 1998; Pradal *et al.* 2009).

We used Python to create links among the three model components because it is a scripting language with a clear and readable syntax. It has a native object-oriented approach, and has an exception-based orientation (Python 2010). It provides extensive standard libraries and many third-party libraries that are freely usable and distributable (Nilsen 2007; Pradal *et al.* 2009). Existing models written in system programming languages can be wrapped into Python libraries using tools like SWIG and F2py without heavy modifications. Dedicated tasks, such as modelling, data management, data analysis and presentation of the spatial and temporal data, can be implemented with the Python libraries.

Used Python libraries

Several Python libraries, written in system programming languages, provide functions needed for development of Taihu PPS (e.g. spatial data manipulation, analysis and visualization). Most libraries are free and open source with the freedom to run the program, to study and adapt the program to one's own needs, to redistribute the program and to improve the program (Steiniger & Hay 2009). This section highlights the critical libraries used in Taihu PPS.

The GDAL is a geographical data access component (Luis 2007). It can be used to read, manipulate and write over 50 raster file formats, including PCRaster and GeoTIFF used in Taihu PPS (GDAL 2010). SWIG was used to link GDAL to Python.

The hydrodynamic-phytoplankton model covers both spatial and temporal processes. Although most GIS software packages contain an extensive set of spatial operators, they are not tailored to dynamic spatial modelling and lack temporal functions (Karssenberg & Jong 2005). The PCRaster modelling framework fills this gap and is tailored to spatial dynamic modelling. The software is free but not open source. The map algebra operations of the PCRaster framework are wrapped in a free library (PCRaster Python), which can be used for spatial and temporal processes in two and three dimensions (Karssenberg *et al.* 2007). The operations are classified into four groups: (1) point or local functions; (2) direct neighbourhood or focal functions; (3) entire neighbourhood functions and (4) functions with a neighbourhood defined by a given topology (Schmitz *et al.* 2009; Karssenberg *et al.* 2010). These operations are useful for processes in Taihu PPS. For example, a local drain direction network map is created with a simple Python script as follows:

$$\text{lddMap} = \text{lddcreate}(\text{dem}, \ \text{arg}_1, \ \ldots, \text{arg}_N)$$

where dem is the digital elevation model, arg_i ($i = 1, 2, \ldots, N$)

Table 1 | Interfaces of the phytoplankton kinetics component, the phytoplankton transport component and the hydrodynamic component

```
# Phytoplankton kinetics component.
class PhyKinetics ( ):
  def__init__ (self, T, DP, DN, Chl, PAR):
    # T, water temperature; data type, PCRaster map.
    # DP, dissolved phosphorus; data type, PCRaster map.
    # DN, dissolved nitrogen; data type, PCRaster map.
    # Chl, initial chlorophyll-α concentration; data type, PCRaster map.
    # PAR, photosynthetically active radiation; data type, PCRaster map.
    …
    return
  def CalPhyGrowth (self):
    …
    return ChlGrowth
    # ChlGrowth, chlorophyll-α concentration; data type, PCRaster map.
  def CalPhyLoss (self):
    …
    return ChlLoss
    # ChlLoss, chlorophyll-α concentration; data type, PCRaster map.

# Phytoplankton transport component.
class PhyTran (Chl, u, v):
  def __init__ (self, Chl, u, v):
    # Chl, initial chlorophyll-α concentration; data type, PCRaster map.
    # u,v, initial water velocity (u,v); data type, PCRaster map.
    …
    return
  def CalPhyTran (self):
    …
    return ChlTran
    # ChlTran, chlorophyll-α concentration; data type, PCRaster map.

# Hydrodynamic component.
class Hydro ( ):
  def __init__ (self, WS, WD, WL, u, v, flows):
    # WS, wind speed; data type, PCRaster map.
    # WD, wind direction; data type, PCRaster map.
    # WL, water level; data type, PCRaster map.
    # u,v, initial water velocity (u,v); data type, PCRaster map.
    # flows, [[river1, discharge],…[riverN, discharge]]; data type, array.
    …
    return
  def CalWaterFlow (self):
    …
    return u1, v1
    # u1,v1, water velocity (u1,v1); data type, PCRaster map.
```

are the arguments to derive the local drain direction network map.

The NumPy library is widely used in many scientific packages for basic linear algebra objects and matrices (Oliphant 2006). PCRaster Python provides conversion functions between maps and matrices (Karssenberg *et al*. 2007). Thus, some operations unavailable in the PCRaster Python library can be performed in NumPy.

A visual component (Matplotlib) was used for flexible raster output, real-time plotting and basic 3D graphics (Hunter 2007). It also offers some basic GIS functions required for spatial data visualization.

Modelling tools

The external libraries mentioned above (Nilsen 2007) make Python well suited to develop flexible modelling tools for data manipulation, statistics and on-screen visualization. A data viewer, a profile tool and a database management tool have been developed.

As the model components concentrate on spatial and temporal processes, a dynamic display of all elements in the models would appeal to the users. Therefore, a data viewer was developed for observed data and model results (Figure 4(c)). It facilitates visualization of a single map, two maps for comparison and time series maps. A toolbar of the data viewer allows: (a) manipulation of geographical data (e.g. zooming, panning and statistical analysis), (b) setting map display styles and colour, (c) map export as images and animations, and (d) simple comparisons between observed data and simulation results.

A profile tool was developed for displaying cell values in time series maps. A database management tool handles input and output data, including field data and remote sensing data. Operations of the data in the database include import, display and export.

Input and output

Taihu PPS opens with the user interface of Figure 4(a) and loads the input data as shown in Figure 4(b). The input configurations can be saved as a file. Thus other users can use it without repeating the configurations. Visualization of the model results (Figure 4(c)) is controlled by the tree view in Figure 4(a).

To demonstrate the workflow and performance of Taihu PPS, we carried out a 3-day (May 26–29, 2008) simulation for this lake. The model inputs include (a) the initial vertically averaged chlorophyll-α concentration, (b) water temperature, (c) dissolved phosphorus and nitrogen, and (d) meteorological data (i.e. wind conditions and cloud cover) during the modelling period. Spatial distributions of the vertically averaged chlorophyll-α concentration, water temperature, dissolved phosphorus and nitrogen were obtained from the interpolated results of measured data (Figure 5). Spatial heterogeneity of the meteorological data was neglected.

When run on a HP xw6600 workstation, Taihu PPS took 78 s for a simulation of a 3-day period. The targeted output is the chlorophyll-α distribution as a measure of phytoplankton biomass. The simulated chlorophyll-α concentrations from May 27 to 29, 2008, are given in Figure 6(a)–(c). The observed chlorophyll-α on May 29, 2008, is given (Figure 6(d)) to compare with the simulation result (Figure 6(c)).

The coupled model simulated distribution of phytoplankton biomass well. Under the wind conditions of the first two days (Figure 6), phytoplankton was transported into Zhushan Bay and resulted in an increase of chlorophyll-α concentration in Zhushan Bay (Figure 6(a)–(b)). This is in accordance with the observed values (Figure 6(d)). The decreasing trend of phytoplankton biomass in most areas of the lake due to the relatively high cloud cover was also captured. Further details about the model performance can be found in Huang *et al*. (2011). This coupled model has already been applied to support decision making (Kong *et al*. 2009).

DISCUSSION

Different from other strategies for model integration (e.g. Warner *et al*. 2008; Cerco *et al*. 2010; Holzworth *et al*. 2010), we used a scripting language to integrate three existing lake models and implement a lightweight and appealing graphical user interface for data exchange between these models. To do so, we compiled the hydrodynamic model into a Python

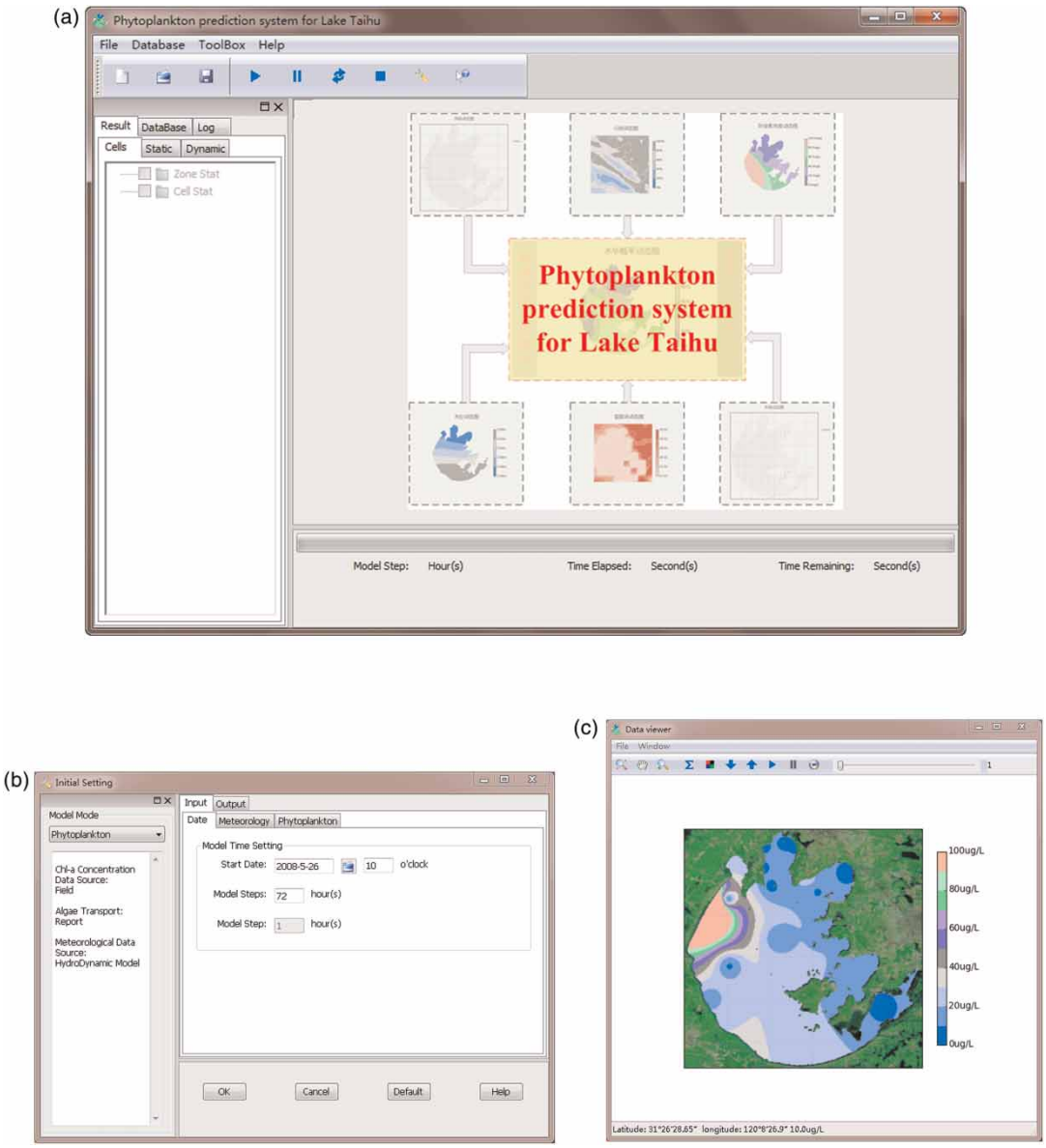

Figure 4 | User interfaces of Phytoplankton Prediction System for Lake Taihu including: (a) main interface, (b) graphical user interface for configuration of the inputs, and (c) the data viewer for displaying hourly simulated chlorophyll-α distribution of Lake Taihu from 10:00 h on May 26, 2008, to 10:00 h on May 29, 2008.

library and designed the interfaces with Python, in a manner that was similar to linkable components (Gregersen *et al.* 2007) and the design-by-interface pattern (Holzworth *et al.* 2010). We also developed functions in *Toolbox.py* (mentioned in the architecture of Taihu PPS) for model integration. Therefore, the approach we took is general and capable of coupling components together without tedious programming. This capability stems from the following advantages of Python in model integration.

Firstly, skilful programming is not required when using Python for model integration. Compared with system programming languages, such as C, C++ and FORTRAN, Python is easier to learn (Karssenberg *et al.* 2007). Therefore, modellers do not have to be professional computer programmers. This feature benefits scientific programming for non-computer scientists.

Secondly, Python is compatible with many other programming languages (Galiano *et al.* 2010). Researchers

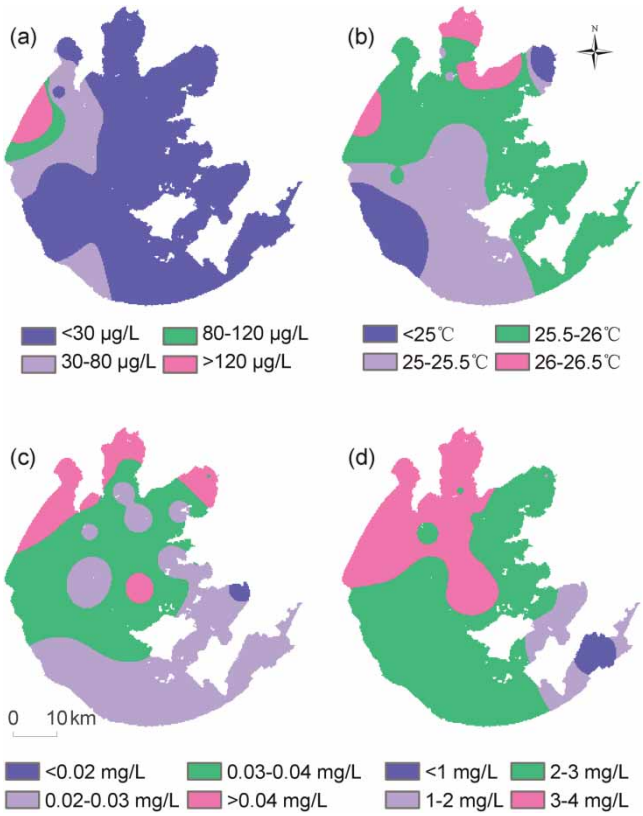

Figure 5 | Model inputs for a 3-day (May 26–29, 2008) simulation of the phytoplankton dynamics in Lake Taihu. (a) The vertically averaged chlorophyll-α concentration; (b) water temperature; (c) dissolved phosphorus; (d) dissolved nitrogen.

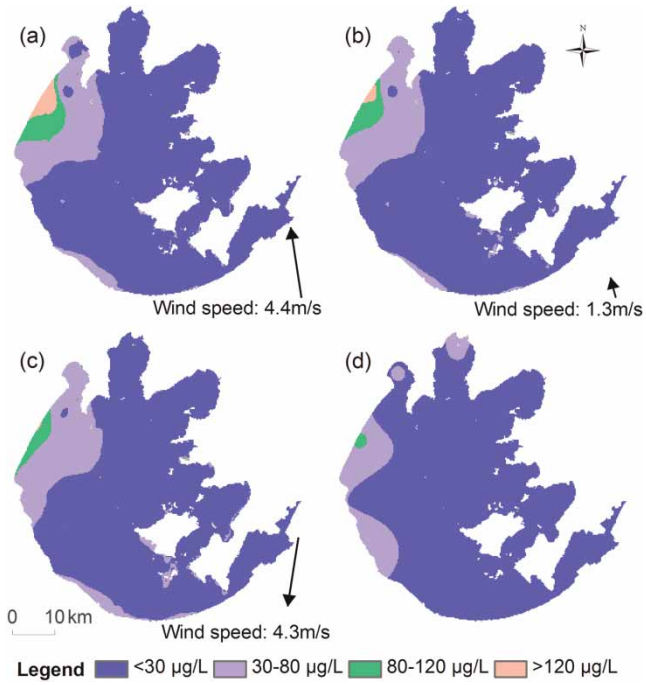

Figure 6 | Time series chlorophyll-α concentration of the simulation results and field data in Lake Taihu. (a) Simulation result for May 27, 2008; (b) simulation result for May 28, 2008; (c) simulation result for May 29, 2008; (d) field data collected on May 29, 2008.

with different backgrounds can develop their model components in a language they are familiar with, and can integrate these components with Python. For instance, we compiled the hydrodynamic model into a Python library and redesigned its interface to link it to the phytoplankton transport model. Thus, the hydrodynamic model component could be easily replaced by an updated version without changing the model code.

Thirdly, Python is free and open source software accompanied by a wide range of free libraries. Many of the geospatial functions (e.g. map algebra, array operations, handling geospatial data and data visualization), required for spatial and temporal modelling, are available as open source GIS libraries (Jolma *et al*. 2008). These GIS libraries manipulate 2D and 3D entities efficiently and are thus suitable for geospatial modelling (Jolma *et al*. 2008; Steiniger & Hay 2009). These software packages are commonly written in system programming languages because they have a short execution time. The libraries are generally implemented by different groups or researchers, and updated libraries with enhanced functions could be accessed freely by model integration specialists. This feature is particularly useful for organizations or individuals with limited resources for programming.

Finally, Taihu PPS developed in Python is independent of other software packages and can be used as stand-alone software. This feature avoids compatibility problems when a new version of the supporting software is released. Furthermore, it avoids the dependency on the developers for support and future maintenance (Roberts *et al*. 2010). In addition, the use of external software requires more experienced users (Matthies *et al*. 2007). Taihu PPS is compiled into binary files, which runs without other third-party libraries. It is therefore easily installed and operated by users with little experience in computer programming.

We conclude that the development of Taihu PPS was successful because of the above-mentioned advantages of Python. However, some potential weaknesses should be

mentioned. In the first place, the level of support and maintenance of free software varies according to the interest and availability of the original authors (Roberts *et al.* 2010). Second, as Python is not tailored to environmental modelling, the requirement to have some basic knowledge of computer programming in model integration is still a challenge for many modelling specialists. Promising approaches to address this challenge are special tools for spatial modelling. For instance, the PCRaster software provides a modelling framework in Python that: (a) provides a framework for stochastic dynamic modelling; (b) contains components for data assimilation; and (c) integrates a calibration toolbox using Genetic Algorithms (Karssenberg *et al.* 2010). These features imply that the modellers can develop models with a few lines of Python code. Third, the model components save outputs (e.g. water flows) as data files for calibration and communication with other components. This exchange strategy between model components wastes time on import, export and conversion of data. Moreover, the amount of data files that are produced makes it a challenge to manage them. This is unacceptable when many model components are included in Taihu PPS. For high-speed communication between model components, in-memory data sharing is a better alternative after the model components are fully verified.

To date, Python is limited to integration of a small number of models. For a large number of models, an approach like the open modelling interface (OpenMI) is promising (Gregersen *et al.* 2007). The development process of Taihu PPS is now still at Level I in Argent's four-level process for developing environmental models (Argent 2004). No attempt is made to make it a universal tool for modelling. A 'build for today' philosophy is adopted to keep Taihu PPS simple and useful. The current development of Taihu PPS is focused on advanced tools for spatial and temporal statistics, 3D visualization and model calibration. Further effort might also focus on a link with a watershed model to provide nutrient and sediment inputs.

As a final remark we would like to note that although Python has the above-mentioned advantages, it may not be as efficient as a suitable framework. It is therefore not our intention with this paper to devalue the importance and necessity of the framework approach and other approaches for model integration. Instead, we would like to provide another choice for model integration when no suitable framework is available in a specific field.

CONCLUSIONS

This paper illustrates the use of an object-oriented scripting language, Python, for a loose and effective coupling of three lake models implemented in different programming languages. Python uses its libraries for map algebra, array operations and visualization. These components are implemented as separate modules, which could be developed by field-specific experts. Python acts as an integrative tool to link the components. The approach has a number of benefits including availability of rich libraries for spatial and temporal representation of lake ecosystem processes, the modular software architecture, open licensing and short execution time. These strengths facilitate its application in different scientific fields.

To learn about the benefits and challenges of using Python in model integration, we developed Taihu PPS for a short-term prediction of phytoplankton distribution in Lake Taihu. Taihu PPS is still far from a mature stage. Further improvements will be carried out to develop tools for advanced statistics, 3D visualization and data assimilation. Updated versions of the Python libraries could contribute to these targets. On the basis of our experiences with developing Taihu PPS, we are convinced that Python is promising for combining different models and deserves more attention from the modelling community.

ACKNOWLEDGEMENTS

We would like to thank the editor and two anonymous reviewers for their useful suggestions. The project was supported financially by the National Basic Research Program of China (Nos. 2008CB418106, 2012CB417006) and the Sino-German cooperation project of 'Assessment of Freshwater Ecosystem under Global Change (EcoChange)'. The authors would like to thank Taihu Laboratory for Lake Ecosystem Research for providing the data for the

simulation example. Special thanks to Brian Moss for improving the manuscript.

REFERENCES

Argent, R. M. 2004 An overview of model integration for environmental application – components, frameworks and semantics. *Environ. Modell. Softw.* **19**, 219–234.

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

Cerco, C. F., Tillman, D. & Hagy, J. D. 2010 Coupling and comparing a spatially- and temporally-detailed eutrophication model with an ecosystem network model: an initial application to Chesapeake Bay. *Environ. Modell. Softw.* **25**, 562–572.

Cheng, W., Wang, C. & Zhu, Y. 2006 *Taihu Basin Model.* 1st edition. HoHai University Press, Nangjing, China (in Chinese).

Galiano, V., Migallón, H., Migallón, V. & Penadés, J. 2010 PyPnetCDF: a high level framework for parallel access to netCDF files. *Adv. Eng. Softw.* **41**, 92–98.

GDAL, Geospatial Data Abstraction Library 2010 Available from: http://www.gdal.org (accessed November 2010).

Govind, A., Chen, J. M., Margolis, H., Ju, W., Sonnentag, O. & Giasson, M.-A. 2009 A spatially explicit hydro-ecological modeling framework (BEPS-TerrainLab V2.0): model description and test in a boreal ecosystem in Eastern North America. *J. Hydrol.* **367**, 200–216.

Gregersen, J. B., Gijsbers, P. J. A. & Westen, S. J. P. 2007 OpenMI: Open modelling interface. *J. Hydroinform.* **9**, 175–191.

Guo, L. 2007 Doing battle with the green monster of Lake Taihu. *Science* **317**, 1166.

Hamilton, D. P. & Schladow, S. G. 1997 Prediction of water quality in lakes and reservoirs. Part I – Model description. *Ecol. Model.* **96**, 91–110.

Hammond, M. & Denn, R. 2000 *Python Programming on WIN32.* 1st edition. O'Reilly & Associates, Inc., Sebastopol.

Holzworth, D. P., Huth, N. I. & de Voil, P. G. 2010 Simplifying environmental model reuse. *Environ. Modell. Softw.* **25**, 269–275.

Hu, W., Jørgensen, S. E. & Zhang, F. 2006 A vertical-compressed three-dimensional ecological model in Lake Taihu, China. *Ecol. Model.* **190**, 367–398.

Huang, J., Gao, J. & Hörmann, G. 2011 Hydrodynamic-phytoplankton model for short-term forecasts of phytoplankton in Lake Taihu, China, *Limnologica*.

Hunter, J. D. 2007 Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95.

Jolma, A., Ames, D. P., Horning, N., Mitasova, H., Neteler, M., Racicot, A. & Sutton, T. 2008 Free and open source geospatial tools for environmental modelling and management. In: *Developments in Integrated Environmental Assessment* (A. J. Jakeman, A. A. Voinov, A. E. Rizzoli & S. H. Chen, eds.). Elsevier, Amsterdam, pp. 163–180.

Karssenberg, D. 2002 The value of environmental modelling languages for building distributed hydrological models. *Hydrol. Process.* **16**, 2751–2766.

Karssenberg, D. & Jong, K. D. 2005 Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. *Int. J. Geogr. Inf. Sci.* **19**, 559–579.

Karssenberg, D., Jong, K. D. & Kwast, J. V. D. 2007 Modelling landscape dynamics with Python. *Int. J. Geogr. Inf. Sci.* **21**, 483–495.

Karssenberg, D., Schmitz, O., Salamon, P., Jong, K. D. & Bierkens, M. F. P. 2010 A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. *Environ. Modell. Softw.* **25**, 489–502.

Kong, F., Ma, R., Gao, J. & Wu, X. 2009 The theory and practice of prevention, forecast and warning on cyanobacteria bloom in Lake Taihu. *J. Lake. Sci.* **21**, 314–328.

Kraft, P., Multsch, S., Vache, K. B., Frede, H. G. & Breuer, L. 2010 Using Python as a coupling platform for integrated catchment models. *Adv. Geosci.* **27**, 51–56.

Lam, D., Leon, L., Hamilton, S., Crookshank, N., Bonin, D. & Swayne, D. 2004 Multi-model integration in a decision support system: a technical user interface approach for watershed and lake management scenarios. *Environ. Modell. Softw.* **19**, 317–324.

Luis, J. 2007 Mirone: a multi-purpose tool for exploring grid data. *Comput. Geosci.* **33**, 31–41.

Matthies, M., Giupponi, C. & Ostendorf, B. 2007 Environmental decision support systems: current issues, methods and tools. *Environ. Modell. Softw.* **22**, 123–127.

Mooij, W. M., Trolle, D., Jeppesen, E., Arhonditsis, G., Belolipetsky, P. V., Chitamwebwa, D. B. R., Degermendzhy, A. G., DeAngelis, D. L., De Senerpont Domis, L. N., Downing, A. S., Elliott, J. A., Fragoso Jr., C. R., Gaedke, U., Genova, S. N., Gulati, R. D., Håkanson, L., Hamilton, D. P., Hipsey, M. R., 't Hoen, J., Hülsmann, S., Los, F. H., Makler-Pick, V., Petzoldt, T., Prokopkin, I. G., Rinke, K., Schep, S. A., Tominaga, K., Van Dam, A. A., Van Nes, E. H., Wells, S. A. & Janse, J. H. 2010 Challenges and opportunities for integrating lake ecosystem modelling approaches. *Aquat. Ecol.* **44**, 633–667.

Nilsen, J. K. 2007 MontePython: implementing quantum monte carlo using python. *J. Comput. Phys. Commun.* **177**, 799–814.

Oliphant, T. E. 2006 *Guide to NumPy*. Available from: http://www.trelgol.com (accessed November 2010).

Ousterhout, J. 1998 Scripting: Higher level programming for the 21st century. *IEEE Computer* **31**, 23–30.

Oxley, T., McIntosh, B. S., Winder, N., Mulligan, M. & Engelen, G. 2004 Integrated modelling and decision-support tools: a Mediterranean example. *Environ. Modell. Softw.* **19**, 999–1010.

Pradal, C., Boudon, F., Nouguier, C., Chopard, J. & Godin, C. 2009 PlantGL: a Python-based geometric library for 3D plant modelling at different scales. *Graph. Models* **71**, 1–21.

Python, Python Software Foundation 2010 Available from: http://python.org (accessed November 2010).

Qin, B., Xu, P., Wu, Q., Luo, L. & Zhang, Y. 2007 Environmental issues of Lake Taihu, China. *Hydrobiologia* **581**, 3–14.

Rahman, J. 2004 Tarsier and ICMS: two approaches to framework development. *Math. Comput. Simul.* **64**, 339–350.

Riley, M. J. & Stefan, H. G. 1988 Minlake: a dynamic lake water quality simulation model. *Ecol. Model.* **43**, 155–182.

Roberts, J. J., Best, B. D., Dunn, D. C., Treml, E. A. & Halpin, P. N. 2010 Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++. *Environ. Modell. Softw.* **25**, 1197–1207.

Saenz, J., Zubillaga, J. & Fernandez, J. 2002 Geophysical data analysis using Python. *Comput. Geosci.* **28**, 457–465.

Schmitz, O., Karssenberg, D., van Deursen, W. P. A. & Wesseling, C. G. 2009 Linking external components to a spatio-temporal modelling framework: coupling MODFLOW and PCRaster. *Environ. Modell. Softw.* **24**, 1088–1099.

Steiniger, S. & Hay, G. J. 2009 Free and open source geographic information tools for landscape ecology. *Ecol. Inform.* **4**, 183–195.

van Kouwen, F., Schot, P. P. & Wassen, M. J. 2008 A framework for linking advanced simulation models with interactive cognitive maps. *Environ. Modell. Softw.* **23**, 1133–1144.

Voinov, A. & Cerco, C. 2010 Model integration and the role of data. *Environ. Modell. Softw.* **25**, 965–969.

Warner, J. C., Perlin, N. & Skyllingstad, E. D. 2008 Using the model coupling toolkit to couple earth system models. *Environ. Modell. Softw.* **23**, 1240–1249.

Wu, X., Kong, F., Chen, Y., Qian, X., Zhang, L., Yu, Y., Zhang, M. & Xing, P. 2010 Horizontal distribution and transport processes of bloom-forming *Microcystis* in a arge shallow lake (Taihu, China). *Limnologica* **40**, 8–15.

First received 14 February 2011; accepted in revised form 11 July 2011. Available online 7 November 2011