Jun 25th, 9:00 AM - 10:20 AM

A Simplified Approach for Water Resources Web Processing Services (WPS) Development

Xiaohui Qiao
Brigham Young University, xiaohui.qiao@byu.edu

Daniel P. Ames
Brigham Young University, dan.ames@byu.edu

Zhiyu Li
Brigham Young University, zhiyu.li@byu.edu

E. James Nelson
Brigham Young University, jimn@byu.edu

Nathan R. Swain
Aquaveo, nswain@aquaveo.com

iEMSs International Congress on Environmental Modelling and Software

9th International Congress on Environmental Modelling and Software
Fort Collins, Colorado, USA, Mazdak Arabi, Olaf David, Jack Carlson, Daniel P. Ames (Eds.)
https://scholarsarchive.byu.edu/iemssconference/2018/

A Simplified Approach for Water Resources Web Processing Services (WPS) Development

Xiaohui Qiao[a], Zhiyu Li[a], Daniel P. Ames[a], E. James Nelson[a], Nathan R. Swain[b]
a Civil and Environmental Engineering, Brigham Young University, Provo, UT, USA (xiaohui.qiao@byu.edu, zhiyu.li@byu.edu, dan.ames@byu.edu, jimn@byu.edu)
b Aquaveo, LLC, Provo, UT, USA (nswain@aquaveo.com)

Abstract: Developing a complex water resources modelling web application can be a daunting task that requires integration of various models and data sources with ever-changing internet technologies. Service-oriented architecture (SOA) has been shown to be useful for building complex modelling workflows. However, compared with other types of web services such as for data delivery and mapping, the implementation of web processing services (WPS) for water resources modelling and data analysis is not very common. Indeed, tools to simplify the development and deployment of WPS for general modelling cases are lacking. We will present the development and testing of a ready-to-use WPS implementation called Tethys WPS Server, which provides a formalized way to expose web application functionality as standardized WPSs in alongside an app's graphical user interfaces. Our WPS server is Python-based and is created on Tethys Platform by leveraging PyWPS. A case study is provided to demonstrate how web app functionality(s) can be exposed as WPS using our open source package, and show how these WPSs can be coupled to build a complex-modelling app. The advantages of Tethys WPS Server includes: 1) lowering the barrier to OGC WPS development and deployment, 2) providing web services-based access of apps, 3) improving app interoperability and reusability, and facilitate complex modelling implementation.

***Keywords**:* Environmental modelling; Web app interoperability; Web processing service; PyWPS; Tethys Platform

1 Introduction

Chaining interoperable model components is gaining momentum for modelling because such a chain can potentially answer more questions than the individual models alone (Castronova et al., 2013; Dubois et al., 2013). Generally, we can achieve model interoperability by sharing input and output files, by directly rewriting models into a single software system, or by establishing software architecture principles that facilitate the coupling of independent models (Belete et al., 2017; Granell et al., 2010). In the last approach, models are written in a modular way, in which each model performs an isolated task while the whole workflow addresses a much broader problem. This approach enables each model to remain as flexible, extensible, and reusable as possible (Argent et al., 2006; Castronova et al., 2013; Schaeffer, 2008). However, integrating multidisciplinary heterogeneous models still involves barriers such as different programming languages, operating platforms, and user interfaces. Service-Oriented Architecture (SOA), which is a standards-based, loosely-coupled, and service-oriented approach emphasizes the decomposition of a system into functional components that communicate via web services (Castronova & Goodall, 2010), has been widely used to integrate models and build scientific workflows (Bosin et al., 2011; Lin et al., 2009; Martin Klopfer, 2009; Nativi et al., 2015; Yue et al., 2016; Zhao et al., 2012). It has been shown that SOA-based applications can address the issues of data accessibility and service interoperability for environmental models (Granell et al., 2010; Zhao et al., 2012). Also, it has been demonstrated that SOA has significantly improved dealing with complex problems by making data and models available on the Internet and coupled via interoperable services (Skøien et al., 2013; Yang et al., 2009).

The Open Geospatial Consortium (OGC) has promulgated the Web Processing Service (WPS) specification, which has been demonstrated as an efficient technology for publishing geospatial processes and constructing integrated model chains (Castronova et al., 2013; Schaeffer, 2008; Tan et al., 2015). Compared with web services for water data distribution (e.g. CUAHSI HIS, HydroShare) and also for web mapping (e.g. USGS, Geoportal), the implementation of WPS for environmental modelling and data analysis is still not particularly common; existing work is more often developed for specific workflows rather than general modelling cases. This lack may be due, in part, to the relatively high barrier associated with developing and deploying web services-based models. We recognize that these processes and workflows can and ought to be exposed via web services in addition to graphical user interfaces (GUI) for optimal flexibility. By so doing, end users will have the flexibility of web services-based access in addition to GUI-based access. However, given the lack of simplified tools for WPS development and deployment, exposing app functionality as a web processing service can be a daunting task; additionally, if developers are left to their own convictions, it is prone to the proliferation of new and random application programming interface (API) definitions. This is symptomatic of the lack of any formalized method for applying WPS on top of an existing web app development framework.

In this paper, we aim to address this lack of a ready-to-use WPS implementation for environmental modelling by developing and testing an OGC WPS system on the Tethys Platform. We elected to use Tethys Platform as our development environment because it is open source and provides a complete and relatively uncomplicated environment for web-based environmental modelling applications development. It has been demonstrated that Tethys Platform can successfully lower the barrier to developing web apps in the environmental domain (Swain et al., 2016). Our goal is to generate an approach of simultaneously exposing web app functionality(s) as WPS when developing a web app. In this way, the barrier of WPS development can be overcome as well. In addition, implementing WPS on Tethys Platform can improve apps' interoperability and reusability and facilitate complex modelling implementation, including, for example, chaining apps for integrated modelling.

The remainder of this paper is organized as follows. A description of our Tethys WPS Server design approach is presented in Section 2. We also present the design of a "HydroProspector" hydrological modelling system to demonstrate how Tethys WPS Server simplifies and enables environmental workflow modelling. The results of this work are described in Section 3. A detailed discussion on the benefits of this study in environmental research and how it successfully lower the barrier to WPS development is presented in Section 4. Section 5 describes suggestions and opportunities for future work in this area.

2 Methods

2.1 Tethys WPS Server design

To achieve our research goal, the first step is to select an existing software product to provide server-side implementation of OGC WPS interface specification. The major reason of directly leveraging an existing OGC WPS implementation project instead of creating a new one from scratch is that we can rapidly implement a relatively mature WPS server. As Tethys Platform is a Python-based and Django-powered web framework (Jones et al., 2014), the WPS implementation software should first support Python scripting. Doing so can facilitate code debugging when developing WPS processes as well as enable Tethys app functions to be converted to WPS processes without modification. In this study, PyWPS (2009) is chosen for the WPS server implementation. PyWPS is a Python-based software which provides a framework that facilitates developers to publish server-side Python applications as web services conforming to OGC WPS standard.

The general design is to implement the WPS server as a plugin of Tethys Platform, which has no effect on other components and functionalities of Tethys Platform. The benefits include (1) leave apps developed based on former versions of Tethys Platform unaffected and (2) make exposing WPS a flexible option, which means the app can execute normally with or without exposing internal functions as WPS processes.

The fundamental purpose of Tethys WPS Server (shown as Figure 1) is to provide a simple way to publish included function(s) of a Tethys web app as OGC WPS in addition to GUI objects. Each Tethys

app typically contains a workflow to provide a solution for one specified task or several tasks. The workflow can be a function or a set of functions, such as an app containing a hydrologic model that performs watershed delineation with a user-defined outlet and then predicts the runoff within this watershed area with latest rainfall forecasts. A Tethys app project is organized using a software development pattern called Model View Controller. The Model refers to storage and retrieval of the data used in a web app. The View is user interface to show buttons, charts, maps and results. The Controller handles computations and execution logic. With a friendly GUI, a Tethys app receives input datasets from users, performs computation in the backend controller, and returns outputs to the front. The core part of a Tethys app is its controller functions, which are implemented in Python. Just as Tethys app, the mechanism of WPS also includes accepting input parameters from a client, executing process in the backend, and returning outputs to the client. Hence, it is theoretically feasible to convert Tethys app Python functions to WPS processes. The primary difference between Tethys apps and WPS processes is each Tethys app defines its specific input and output formats, while the inputs and outputs of WPS processes are both uniformly defined in XML format. The primary benefit of exposing app functionalities as OGC WPS is the WPS hosted on Tethys WPS Server can be included and reused in the workflow of other Tethys apps and any other third party client that supports OGC WPS.

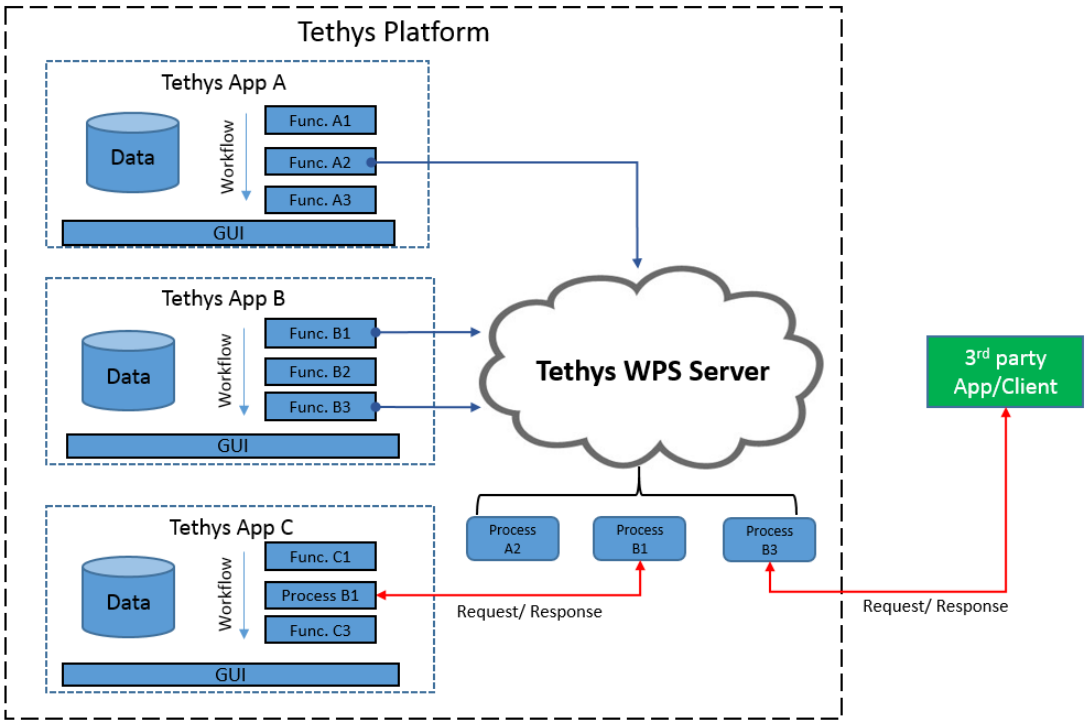

Figure 1. System design of Tethys WPS Server

2.2 Experimental case study design

A service-oriented web application, HydroProspector, is designed and implemented to illustrate that (1) functionalities of a Tethys app can be easily exposed as WPS processes on Tethys WPS Server, (2) services hosted on Tethys WPS Server can be directly included in another Tethys app as components of the workflow, and (3) Tethys WPS Server can signicantly lower the barrier to developing web apps with complex modelling and analysis.

HydroProspector is a reservoir management model which calculates the storage capacity curve of a reservoir to help decision-makers choose the best dam location. The storage capacity curve, which refers to the relationship between dam height and reservoir storage capacity, is important for dam planners, designers and operators (Issa et al., 2017). Figure 2 shows the designed workflow of HydroProspector App. To calculate the storage capacity curve, the first step is delineating watershed based on a dam location. The watershed basin defines the maximum reservoir size and works as a boundary for all following geoprocessing calculations. Next, the storage capacity of the reservoir for different dam heights is calculated and finally, the curve is plotted.

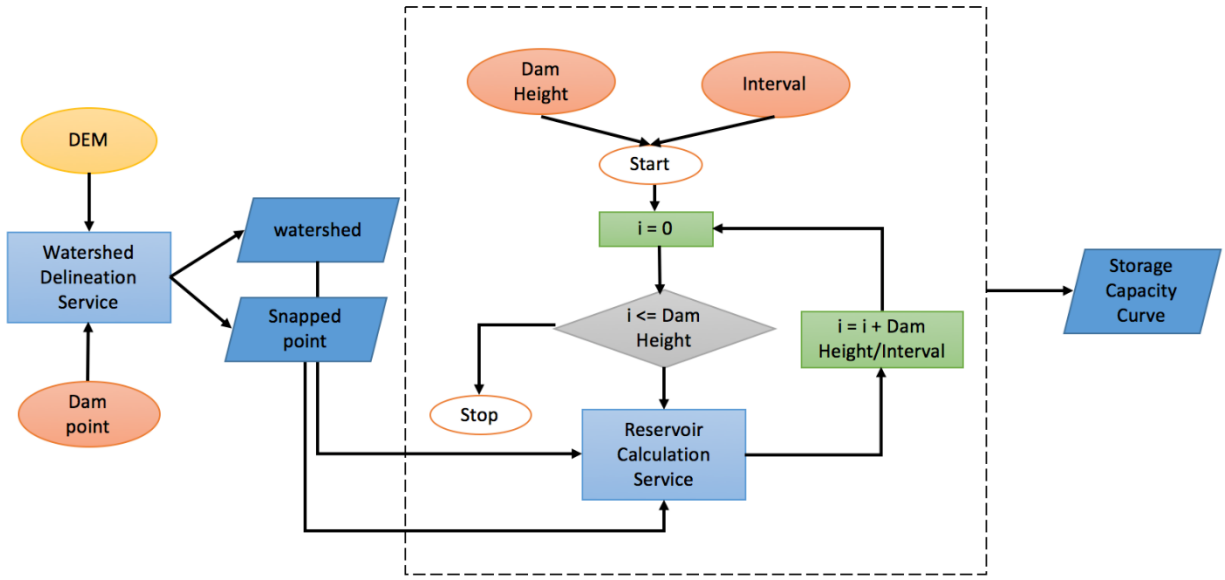

Figure 2. HydroProspector app workflow

There are two WPS processes used in this app: a watershed delineation service accepts coordinates of a point and returns a watershed and the snapped point at nearest stream, a reservoir calculation service calculates reservoir volume with coordinates of a pour point and the target water level at this point. Both these two WPS processes are exposed from well-developed Tethys apps (Watershed Delineation for Dominican Republic App and Reservoir Calculation for Dominican Republic App) and support asynchronous execution. The output of the watershed delineation service is a watershed in GeoJSON format, while one input of the reservoir calculation service is a polygon GeoJSON file. Having variables in the same format enables the connection of these two services. As all services hosted on Tethys WPS Server conform to OGC WPS specification, they can theoretically be easily combined or integrated into a workflow.

3 Results

3.1 Tethys WPS Server results

An API has been developed for exposing Tethys app functions as OGC WPS processes. When a user generates a new Tethys app through Tethys scaffold command, a WPS definition template file is automatically included in the app project. Each WPS process is defined as a PyWPS class containing metadata configuration and a handler method that specifies the execution procedure. Other outside functions defined in the Tethys app can be called in the handler method to directly convert to a WPS process or as part of the WPS process. This design allows a convenient revision of the WPS as they can be simultaneously modified with the app.

A set of command lines are set to help manage WPS processes on Tethys WPS Server, including a “tethys wps list” command for listing all published and unpublished WPS processes, a “tethys wps publish” command for publishing WPS processes, and a “tethys wps remove ”command to remove selected WPS process(s). The benefits of using command lines include simplifying common management tasks related to WPS and allowing app developers to flexibly modify WPS processes without affecting apps’ normal operation.

We choose the open source desktop GIS software Quantum GIS (QGIS) as a WPS client to demonstrate that the WPS processes hosted on Tethys WPS Server are standard to be used by other third party clients that support OGC WPS specification. QGIS is chosen because it is open source and it supports WPS with a WPS client plug-in. Through the WPS client plug-in, QGIS can successfully access Tethys WPS Server and list all services hosted on it. The Watershed Delineation Process WPS and Reservoir Calculation Process WPS work well in QGIS.

3.2 Case study results

We deploy three Tethys web apps at the website (http://tethys.byu.edu/apps): Watershed Delineation for Dominican Republic App, Reservoir Calculation for Dominican Republic App, and HydroProspector for Dominican Republic App. From the first two apps, we respectively exposed a "Watershed Delineation Process" WPS and a "Reservoir Calculation Process" WPS on Tethys WPS Server. The HydroProspector App is developed by coupling these two WPS processes. The HydroProspector app user interface (Figure 3) includes an interactive map to show watershed and reservoir boundaries, along with a chart to show the storage capacity curve of designed dams. The user is required to (1) click on the map to select an outlet point for watershed delineation, and then (2) define a dam height and interval. The app will dynamically draw the storage capacity curve in the chart and show corresponding reservoir boundary with different dam heights. When selecting a data point in the storage capacity curve chart, the corresponding reservoir boundary is shown on the map.

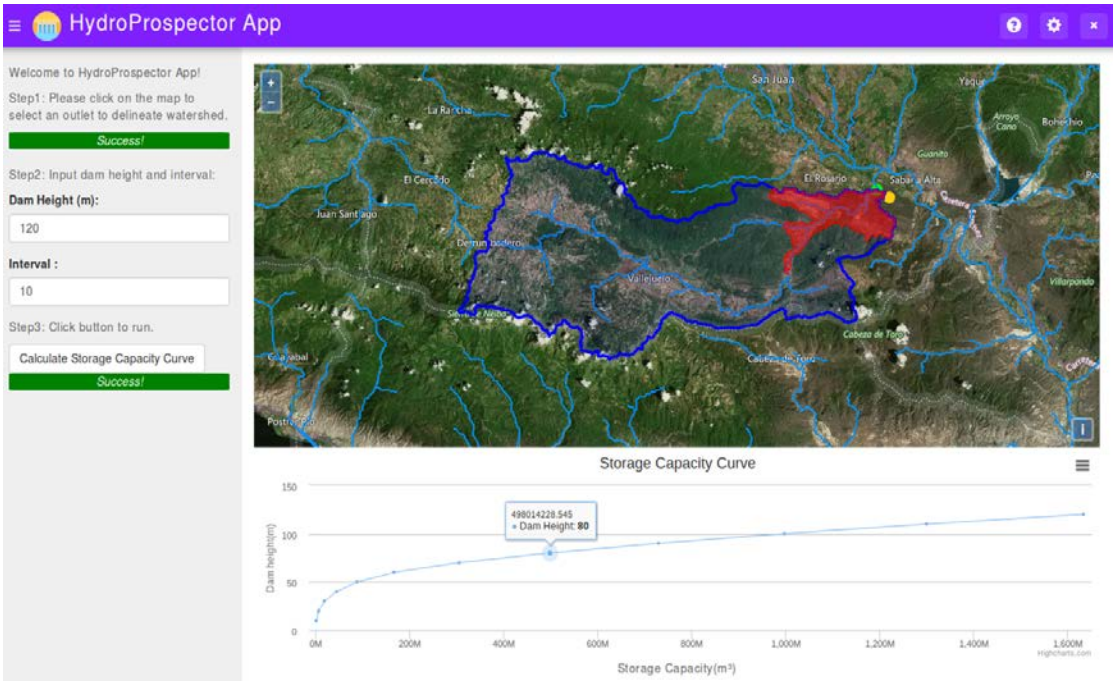

Figure 3. HydroProspector App User Interface

4 Discussion

The motivation of this study is to create a ready-to-use tool to expose web app functionalities to standard OGC WPS, thereby facilitating the development of web apps containing complex environmental models. PyWPS is chosen for server-side OGC WPS implementation because it is Python-based and supports exposing Python scripts. Tethys Platform is chosen as the development environment because of its advantages in developing web-based environmental applications. Tethys Platform provides a complete modelling environment including geoprocessing tools for spatial data, map rendering and visualization, distributed computing, and database management (Jones et al., 2014), which features enable us to quickly establish a web application with limited web development skills. A template file is added to Tethys app project source code for WPS definition, which can call app's functions for converting them into WPS processes. A set of command lines are developed for WPS management. The key outcomes of this work for app developers using the Tethys platform include:

1) Ease of implementation - It is possible to add WPS processes without needing to change existing well-written codes in an existing web app.
2) Easier maintenance - A WPS is based on and connected with a web app, so these services can be debugged, tested, and updated together with the web app, enabling simpler maintenance of the app and services.
3) Avoidance of code duplication – As OGC WPS can be used across applications with different programming language over different platforms, WPS processes hosted on Tethys WPS server

can be used within other Tethys apps, or by third party applications or clients which support OGC WPS specification.

4) Lowering the barrier to environmental OGC WPS development and deployment - Tethys Platform has been demonstrated to successfully lower the barrier to developing and deploying environmental web applications. Tethys WPS Server provides a simple method to expose web app functions as OGC WPS. Therefore, the barrier of developing OGC WPS should be overcome.
5) Facilitating complex environmental web apps development - Implementing complex environmental models to web apps can be a daunting task because complex models are normally long-running processes, which require including task management, asynchronous execution, data storage, and other possible issues related. Instead, decomposing the model to loosely coupled WPS processes can avoid many web developing issues, especially benefits environmental scientists and engineers with limited programming skills. Moreover, a WPS process can contain a series of other WPS processes, which is necessary for environmental modelling as many complex environmental models contain multiple levels. For example, some models may first decompose the water system into atmosphere, land surface, and subsurface, then each component may contain a more detailed process-level service, such as land surface including infiltration, evaporation, runoff, and others, and each process can be decomposed to multiple smaller services.

In addition to creating a Tethys-specific implementation of WPS and demonstrating its utility, this work also serves as general methodological experience and guidance for alternative implementations of WPS, which may not use the Tethys/Django framework. Specifically:

1) We have shown that disaggregation of a complex workflow into multiple modular WPS instances can help avoid overloading a single server and does yield many of the benefits reported by Castronova et al. (2013) and others reported in the introduction section;
2) Others can learn from and emulate our success in deploying GIS-enabled web applications that include both a front-end graphical user interface for end users AND a backend WPS service which can be activated and deactivated through simple server side commands;
3) Our experience with the PyWPS library specifically is highly positive and serves as a recommendation for this library in similar use cases where Python is the primary server-side technology; and
4) Maintaining the integrity of PyWPS (with only minor changes) as a separate package that is linked into our Tethys infrastructure, we gain the benefit of being able to readily upgrade to newer versions of this library that may become available in the future.

We expect that each of these observations and outcomes will have broader application for future environmental web application developers beyond the Tethys user and developer community.
We also note a few challenges that remain with respect to this work. First, functions of an app cannot be converted into WPS automatically without extra-coding. App developers must manually define WPS processes following the API, which also causes some level of code duplication in the app, including request and response data processing. Second, Tethys WPS Server currently is accessible to anyone connected to the Internet without any authorization or authentication requirements. This should be subjected to Tethys Platform's user management and access control. The third challenge is not a shortcoming of Tethys WPS Server, but is a shortcoming of WPS in general and is related to moving large datasets between services over the internet, which can cause network latency. Some methods have been implemented to minimize network latency, such as combining processes that require numerous data transfers into a single service (Goodall et al., 2011), saving inputs/outputs in optimized binary-format such as Network Common Data Form (NetCDF) files (Jiang et al., 2017). That means even though Tethys WPS Server provides an easy environment to develop WPS, there remains a high requirement for users' ability to decompose a complex modelling system into a set of representative services, and design services with appropriate inputs/outputs to reduce network latency.

5. Conclusion and future work

In this study, we have developed a Tethys WPS Server using PyWPS to expose web app functionalities to standard OGC WPS processes. A set of command lines has been set for flexible WPS processes management, including publishing, listing, and removal. A Tethys web application coupling two WPS

processes has been developed to demonstrate the advantages and benefits of Tethys WPS Server for complex environmental modelling systems. Through Tethys WPS Server, one or more functionalities in each Tethys App can be easily exposed as OGC WPS processes, and deployed along with the Tethys app. With the work presented in this study, Tethys WPS Server can lower the barrier to OGC WPS development and deployment, and further improve complex modelling web applications development in the environmental domain. The next step will be to upgrade the Tethys WPS Server from a Tethys internal application to a Django application that can be easily coupled with any Django-based system; to handle the problem of security and user authentication; and to provide more examples and guidance on well-designed modularization and decomposition of complex problems to avoid large data passing problems.

Software Availability

Tethys Platform

- Source code: https://github.com/tethysplatform/tethys
- Home page: http://www.tethysplatform.org/
- License: The BSD 2-Clause open source license

PyWPS

- Source code: https://github.com/geopython/pywps
- Home page: http://pywps.org/
- License: The MIT license

Tethys WPS Server

- Source code of Tethys Platform including Tethys WPS Server: https://github.com/tethysplatform/tethys/tree/tethys_wps
- URL: https://tethys.byu.edu/tethys_wps/?service=WPS&request=GetCapabilities
- License: BSD 2-Clause open source license.

Case Study Apps

Three web apps in the case study can be found at https://tethys.byu.edu/apps. The source code for these three web apps are available on GitHub in the following repositories:

- Watershed Delineation for Dominican Republic App: https://github.com/xhqiao89/watershed_delineation_app
- Reservoir Calculation for Dominican Republic App: https://github.com/xhqiao89/reservoir_calculation_app
- HydroProspector for Dominican Republic App: https://github.com/xhqiao89/hydroprospector_app

Author Contributions and Acknowledgements

Author contributions are as follows: Qiao did the bulk of the code development and writing, Li made significant code contributions, Ames was the primary research advisor and contributed extensively to the writing, Nelson and Swain provided significant intellectual support in terms of the case study and Tethys Platform integration. The authors express their appreciation to Dr. Norman L. Jones, Tethys Steering Committee, and BYU Hydroinformatics Lab for their input which has improved this work.

References

Argent, R. M., Voinov, A., Maxwell, T., Cuddy, S. M., Rahman, J. M., Seaton, S., Braddock, R. D., 2006. Comparing modelling frameworks–a workshop approach. Environmental Modelling & Software, 21(7), 895-910.

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

Bosin, A., Dessì, N., & Pes, B., 2011. Extending the SOA paradigm to e-Science environments. Future Generation Computer Systems, 27(1), 20-31.

Castronova, A. M., & Goodall, J. L., 2010. A generic approach for developing process-level hydrologic modeling components. Environmental Modelling & Software, 25(7), 819-825.

Castronova, A. M., Goodall, J. L., & Elag, M. M., 2013. Models as web services using the open geospatial consortium (ogc) web processing service (wps) standard. Environmental Modelling & Software, 41, 72-83.

Dubois, G., Schulz, M., Skøien, J., Bastin, L., & Peedell, S., 2013. eHabitat, a multi-purpose Web Processing Service for ecological modeling. Environmental Modelling & Software, 41, 123-133.

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.

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

Issa, I. E., Al-Ansari, N., Sherwany, G., & Knutsson, S., 2017. Evaluation and modification of some empirical and semi-empirical approaches for prediction of area-storage capacity curves in reservoirs of dams. International Journal of Sediment Research, 32(1), 127-135.

Jiang, P., Elag, M., Kumar, P., Peckham, S. D., Marini, L., & Rui, L., 2017. A service-oriented architecture for coupling web service models using the Basic Model Interface (BMI). Environmental Modelling & Software, 92(Supplement C), 107-118.

Jones, N., Nelson, J., Swain, N., Christensen, S., Tarboton, D., & Dash, P., 2014. Tethys: A Software Framework for Web-Based Modeling and Decision Support Applications. Paper presented at the 7th International Congress on Environmental Modelling and Software, DP Ames, NWT Quinn, and AE Rizzoli. (Eds.), iEMSs, San Diego, California.

Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z., Hua, J., 2009. A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution. IEEE Transactions on Services Computing, 2(1), 79-92.

Martin Klopfer, I. S., 2009. SANY - an open service architecture for sensor networks.

Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M., & Ochiai, O., 2015. Big Data challenges in building the Global Earth Observation System of Systems. Environmental Modelling & Software, 68, 1-26.

PyWPS, 2009. Python Web Processing Service (PyWPS) (Version 4.0.0). http://pywps.org.

Schaeffer, B., 2008. Towards a transactional web processing service. Proceedings of the GI-Days, Münster.

Skøien, J. O., Schulz, M., Dubois, G., Fisher, I., Balman, M., May, I., & Tuama, É. Ó., 2013. A Model Web approach to modelling climate change in biomes of Important Bird Areas. Ecological informatics, 14, 38-43.

Swain, N. R., Christensen, S. D., Snow, A. D., Dolder, H., Espinoza-Dávalos, G., Goharian, E., Burian, S. J., 2016. A new open source platform for lowering the barrier for environmental web app development. Environmental Modelling & Software, 85, 11-26.

Tan, X., Di, L., Deng, M., Fu, J., Shao, G., Gao, M., Jin, B., 2015. Building an Elastic Parallel OGC Web Processing Service on a Cloud-Based Cluster: A Case Study of Remote Sensing Data Processing Service. Sustainability, 7(10), 14245-14258.

Yang, C.T., Huang, G.L., Huang, F.H., Ho, T.W., Huang, P.H., & Yang, J.M., 2009. SOA-based platform for water resource information exchanging. Paper presented at the 2009 17th International Conference on Geoinformatics.

Yue, S., Chen, M., Wen, Y., & Lu, G., 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.

Zhao, P., Di, L., & Yu, G., 2012. Building asynchronous geospatial processing workflows with web services. Computers & Geosciences, 39, 34-41.