Design and development of a web-based EPANET model catalogue and execution environment

Tylor Bayer*, Daniel P. Ames** and Theodore G. Cleveland**

*Civil and Environmental Engineering, Brigham Young University, Provo, UT, USA; **Civil, Environmental, and Construction Engineering, Texas Tech University, Lubbock, TX, USA

ABSTRACT
Within the field of environmental and hydrologic modelling, there is a growing recognition of the scientific and educational value of sharing both model programs (i.e. the codes that formulate a model) and model instances (i.e. specific input files and model parameterizations). Indeed, numerous cyberinfrastructure tools have been created in recent years to simplify linking, executing, and sharing models. Multiple challenges hinder the adoption of such systems by modelling communities. These can include technological barriers such as installing and managing software and internet systems as well as social barriers such as a culture of not sharing models due to concerns over private or otherwise proprietary information. In some cases, model-sharing platforms are not easily replicated or implemented to fit the needs of a specific modelling community. This paper presents the design and implementation of a model-sharing repository and a model-viewing application, specifically for the EPANET modelling community – the pattern and structure of which could be easily adopted by any modelling community – using existing open source cyberinfrastructure. We used HydroShare as the backend data store for the EPANET model program, model instances, and metadata, and we used the rapid app development capabilities of Tethys Platform framework to create a web-based front-end for the repository and viewer. Results of this experimental work include a functional model repository based on less than 700 lines of code and a light-weight model viewer application that encompasses nearly 100% of the legacy EPANET desktop GUI’s functionality.

1. Introduction
Collaboration among modelling experts and stakeholders can increase the rate at which science advances and, in turn, can increase the resulting benefits (Voinov and Bousquet 2010). Sharing model programmes and model instances is an integral part of this collaboration. As such, the means by which models are shared should be efficient and effective (Chen et al. 2020; Zhang et al. 2019).

Many tools and technologies have been developed to help facilitate sharing of scientific research and engineering application models, model codes, and model instances. Some of these include GitHub (Perkel 2016), FigShare (Singh 2011), CSDMS (Peckham, Hutton, and Norris 2013), OpenGMS (Chen et al. 2020), and HydroShare (Tarbott et al. 2014). Following is a brief description of how these tools can be used to share executable models and model codes.

(1) GitHub – ‘Host and review code, manage projects, and build software alongside 50 million developers’ (github.com). Users can collaboratively work on improving model codes and store model executables.
(2) FigShare – ‘A home for papers, FAIR data and non-traditional research outputs that is easy to use and ready now’ (figshare.com). Researches can store, and discover data in a way that promotes collaboration and reduces duplicated effort.
(3) CSDMS – ‘Community Surface Dynamics Modeling System’ (csdms.colorado.edu). CSDMS creates protocols for open source software, distributes software tools and models, and provides cyberinfrastructure to support these efforts (Jiang et al. 2017; Overeem, Berlin, and Syvitski 2013).
(4) OpenGMS – ‘OpenGMS supports sharing of your modeling and simulation resources for geographic applications’ (geomodeling.njnu.edu.cn). OpenGMS provides collaboration tools for analysing geographic modelling applications (Chen et al. 2020; Zhang et al. 2019).
(5) HydroShare – ‘Online collaboration environment for sharing data, models, and code’ (hydroshare).

CONTACT Daniel P. Ames dan.ames@byu.edu Civil and Environmental Engineering, Brigham Young University, Provo, UT 84602, USA
© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group, on behalf of Nanjing Normal University. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
HydroShare provides a cloud datastore where resources are paired with relevant metadata to enable easy collaboration. HydroShare also provides a powerful API for programmatically fetching data and other resources (Crawley et al. 2017; Khattar et al. 2018; Tarboton et al. 2014).

While there is growing capability to share models and model codes as part of open modelling and simulation for scientific research and engineering, there is a still a sense in the literature that many modellers and modelling communities are not sharing models and model codes (Soranno et al. 2015). Indeed, in light of the recent COVID-19 pandemic, there has been a growing call for openness and transparency of models, model codes, and modelling uncertainty in the medical research field (Barton et al., 2020). Indeed, in spite of the many tools and technologies that have been developed to facilitate sharing of model instances and model codes, there is a barrier to entry because of the complexity of using and deploying these tools. These challenges have been noted in terms of sharing web-based decision support tools (Swain et al. 2016), reusing heterogeneous geo-simulation model resources (Zhang et al. 2019), dealing with social, cultural, and institutional constraints (Nativi, Mazzetti, and Geller 2013), and in the context of using open GIS data in models (Zhu and Yang 2019). Additional research has been done on model sharing in the context of encouraging collaboration and improving research reproducibility (Gan et al. 2020).

We hypothesize that, given a thoughtful and careful integration of existing open source tools and technologies, it is possible to assemble systems for model sharing that lower the barriers to sharing and reusing models. Specifically, for this research, our goal is to investigate the use of free and open source software tools to develop a web-based ecosystem for sharing, viewing, editing, running, and analysing simulation results of model instances of the EPANET water distribution network modelling software. We used HydroShare, described previously, as well as the open source web application system, Tethys Platform (Swain et al. 2016), to host a catalogue of EPANET model instances and a live, web-based EPANET model execution tool to test and explore these models in real time. Figure 1 shows the software architecture using integrating HydroShare and Tethys Platform. The steps shown in the figure are explained in Table 1. The Model Repository and Model Viewer form the bulk of this research and are described in the following sections.

This paper presents the development of the architecture in Figure 1 specifically to support the EPANET model and community. We also intend this work to represent and demonstrate a pattern and structure, which could be easily adopted by any modelling community, using existing open source cyberinfrastructure. HydroShare and Tethys Platform development teams have argued that these tools are useful for, ‘cleaning your desktop by moving your work to the cloud’ and in ‘lowering the barriers to web app

![Figure 1](image-url)
Table 1. Descriptions of numbered steps from web ecosystem architecture diagram.

| Step | Description |
|------|-------------|
| 1    | Repository app sends json request to backend python to get the EPANET model list |
| 2    | Python sends API request to HydroShare through OAuth to get the EPANET model list, including accompanying metadata |
| 3    | HydroShare sends the model list in response to the authenticated request |
| 4    | Python sends the model list response to the front end |
| 5    | The model list is parsed and rendered |
| 6    | When a model is selected to be opened in the viewer app the model ID is sent with the request |
| 7    | Viewer app receives the model ID and sends an json request to backend python to get the respective EPANET model file |
| 8    | Python sends API request to HydroShare through OAuth to get the EPANET model file, including accompanying metadata |
| 9    | HydroShare sends the model file in response to the authenticated request |
| 10   | Python sends the model file response to the front end |
| 11   | The model file is parsed and rendered to be viewed, edited and analysed |

development,’ respectively (Swain et al. 2016; Tarboton et al. 2014). This research tests these statements by implementing these tools in an operational modelling community. Our experience developing tools and capabilities for the HydroShare software stack informs our use of this piece of critical cyberinfrastructure (Crawley et al. 2017; Sadler, Ames, and Livingston 2016). Our experience with developing Tethys Platform web applications has proven invaluable to informing the development of the apps presented here (Evans et al. 2020; Kadlec, Miller, and Ames 2016; Perez et al. 2016; Sadler, Ames, and Livingston 2016).

There are a number of limitations and challenges to model sharing within any modelling community. Some of the barriers described in the aforementioned literature include technological barriers such as installing and managing software and internet systems. Other barriers discussed in the literature include social barriers such as a culture of not sharing models due to concerns over private or otherwise proprietary information. The research presented here is intended to directly address the question of model-sharing platforms not being easily replicated or implemented to fit the needs of a specific modelling community. We recognize that this does not necessarily lower the general technical and social barriers within communities. These challenges certainly provide opportunities for future research.

The remainder of this paper is organized as follows: Section 2 describes our selected cyberinfrastructure, HydroShare and Tethys Platform, in some detail; Section 3 describes our selected simulation model, EPANET, in some detail; Section 4 describes the design and development of an EPANET model repository application for storing and discovering model instances based on Tethys Platform and HydroShare; Section 5 describes the design and development of a model viewing and execution web application for EPANET models based on Tethys Platform; finally, Section 6 includes some discussion and conclusions that summarize this work and look forward to future potential opportunities related to this effort.

2. EPANET

Developed by the United States Environmental Protection Agency (EPA), EPANET is a powerful open-source software suite used throughout public and private industries to model water distribution networks. In the past 5 years alone, there have been over 7,000 references to EPANET in peer review journals. Many of these research efforts tend to optimizing and extending EPANET’s capabilities, making it applicable to a wider range of real-world modelling scenarios. Some highly cited research articles related to EPANET include:

(1) Uniformly Distributed Demand EPANET Extension (Menapace et al. 2018).
(2) Noniterative Application of EPANET for Pressure Dependent Modelling Of Water Distribution Systems (Sayyed, Gupta, and Tanyimboh 2015).
(3) OOPNET: An object-oriented EPANET in Python (Steffelbauer and Fuchs-Hanusch 2015).
(4) Quest for a New Solver for EPANET 2 (Burger et al. 2016).

EPANET models are fairly straightforward and easy to conceptualize. They consist of nodes and links (also known as ‘nodes’ and ‘edges’ in computer science terminology) representing the water distribution network to be analysed (Figure 2). Nodes and links come in various types, each with their own specific set of settings and configurations (Table 2). Besides nodes and links, an EPANET model also includes various global settings. Selected head-loss equation is one of the main settings the model depends on. The EPANET modelling software includes the Hazen-Williams, Darcy-Weisbach, and Chezy-Manning head-loss formulas. Based on the global model settings and hydraulic properties of each component in the network a model simulation can be ascertained and analysed. A sample tutorial model is shown in Figure 3 with only a handful of nodes and links. Many production level models have 80,000 nodes and links or more. As is to be expected, larger models require considerably more time to run, especially if an extended period analysis is desired.

Applications of the EPANET modelling program range from measuring water demands and requirements on
a system, discovering energy inefficiencies in existing water distribution infrastructure, to identifying security risks of contaminant spread through a system and analysing the after-effects of such an event. When new large construction areas are proposed in a city, an impact study of the new water distributions zones on the existing infrastructure is often required. Hydraulic engineers can run extended period simulations of the existing system with the proposed additions and model how the demands at each point in the network change.

EPANET software includes an interactive model editing GUI for Microsoft Windows. The application allows engineers to more easily construct models and visualize the results of a model simulation. Other editors, such as demand curve, flow curve and pattern editors, are also provided to further streamline the creation of EPANET models. This GUI is what we tried to reproduce and improve upon through the EPANET Model Viewer as a lightweight web application presented in this paper.

3. Model repository

3.1. Requirements

The design requirements for this project provided a definitive starting point for development of the resulting software and also served as a guide throughout later stages of development. The requirements for the model repository app were very clear from the beginning and did not change significantly throughout development and were as follows:

1. HydroShare integration as backend data store and authentication vehicle
2. A catalogue for discovering and storing EPANET models
3. Filtering and searching tools to make finding relevant models easier
4. The ability to download models to be used locally or to open models in the EPANET Model Viewer for editing and analysing
(5) A simple enough application structure such that nearly 100% of the code could be reused as a repository for other modelling communities.

HydroShare’s resource discovery page acted as a basic template for the creation of this application as it meets most of the requirements listed above. As the discovery page is intended to be a catalogue for all resource types that exist on HydroShare, the UI/UX is more complicated than was desired for the scope of our design goals. Having a specialized catalogue in the form of a standalone web application for a specific resource type is a useful and powerful tool. It reduces the amount of time to discover the resources and facilitates added features and tools for using them; such as opening a model type resource in another web application that can read and analyse it (see Figure 4). Clicking the ‘Open Model’ button launches the model viewer which supports viewing, editing, and running EPANET models in a web-based model viewer application without any added software installed on their machine.

In addition to these features, we wanted the process of installing both applications on any Tethys portal to be extremely easy. The Tethys App Warehouse enabled this capability (Khattar et al. 2018). Any Tethys portal with an

![Figure 3. Larger model example (about 10,000 nodes and 10,000 pipes).](image)

![Figure 4. EPANET model repository.](image)
App Warehouse can have the EPANET Model Repository and Model Viewer installed on it with a single click.

3.2. Technology

Several open source technologies were used in the creation of the EPANET Model Repository. The key technologies include:

1. Tethys Platform – Rapid web application development and installation on Tethys portals; Python backend and Django front-end; Powerful code scaffolding tools; Built-in social authentication hooks (Figure 5).
2. HydroShare – Backend model instance data and metadata store; REST API plugin for retrieving and storing model instances; Social authentication.
3. jQuery – JavaScript library that simplifies verbose DOM interactions (document object model for HTML) and AJAX requests (client-side asynchronous http request technique).
4. CDN DataTables – jQuery plugin for creating reactive, paginated, filterable and interactive HTML tables (See Figure 4 – showing DataTables UI element).
5. Bootstrap – Industry standard front-end styling toolkit.

3.3. Implementation

Although a basic knowledge of the standard web development languages (HTML, JavaScript, CSS) and Python were required to build the EPANET Model Repository, many of the technologies used are industry standard tools that make implementation much easier and faster.

Enabling HydroShare social authentication was the first step in implementation. Tethys Platform includes a HydroShare REST API Client helper function to facilitate social authentication. Figure 6 illustrates the integration with HydroShare. We first pass the AJAX request object into the Tethys HydroShare authentication helper function and use the returned ‘hs’ object to loop through all model instance resources that have ‘epanet’ as their subject. We add each of these resources with pertinent metadata, such as title, owner and availability information, to an array of models and return it to the front-end JavaScript code to generate the catalogue table. Once the model list is returned, we generate a standard HTML table with the pertinent metadata as columns. Creating an interactive table requires registering our table element as a CDN DataTable through its element id.

The table filters were implemented as checkboxes that augment which models in the model list were available to be displayed through the CDN DataTable plugin. The Owner and Subject filters are created ‘on the fly’ based on the set of owners and subjects of all models in the model list. As there are a set number of availability types (shareable, public, private, discoverable and not shareable) in HydroShare, this filter was hard-coded. The filters’ click events only add models to be displayed from the list that meet their criteria.

We also added some click events to the rows in the table; the main event being opening the model in the EPANET Model Viewer. This is done by redirecting the client to the URL of the EPANET Model Viewer with the HydroShare resource ID appended to the URL as a query parameter.
parameter. Opening models in other web applications that are compatible with EPANET Models could be accomplished the same way as long as the apps are able to implement HydroShare in the backend.

### 3.4. Model repository results

Our primary goal with the EPANET Model Repository was to create a powerful but simple standalone catalogue specifically for EPANET model instances. We intended for the structure to be simple enough that with minimal changes the application could be converted to fulfil the same function for any other modelling community. We believe the EPANET Model Repository meets these goals. With a little over 700 lines of code, most of which being reusable, any data scientist or engineer with minimal web development experience could create a web-based model instance catalogue for the peers in their community to start to share and learn together in one convenient place. Using HydroShare as a backend datastore with its REST Client for storing model instances makes tying into other applications, that can use the model instances, extremely easy. The technology used in the EPANET Model Repository synergizes very well and enables the cohesive web-ecosystem that we sought to create.

In the future, we would like to automate the process of converting the EPANET Model Repository into a repository for a different modelling community. With some careful planning we believe this would not be very difficult.

### 4. EPANET model viewer

#### 4.1. Requirements

Below, are the key features we identified in the end as most important to realizing our research goal of creating a full web ecosystem for EPANET once the model repository app was complete:

1. HydroShare integration as backend data store
2. Authentication through HydroShare
3. A large graphical window for viewing the visual components of an EPANET model
4. Network editing tools including node creation, edge creation and a search tool
5. Additional model option editing tools including secondary graphical editors, such as demand curve, flow curve and pattern editors
6. Ability to run models and view simulation results including time series animation of results

#### 4.2. Technology

Several of the same technologies used in the EPANET Model Repository were also implemented in the EPANET
Model Viewer, however, being a more specialized application, it required the addition of more specialized tools. In addition to Tethys Platform, HydroShare, jQuery, and Bootstrap (as described in Section 3.2) key technologies used in the EPANET Model Repository include:

1. EPANETTOOLS – Python wrapper for running EPANET models; Installed with pip; Requires a C compiler.
2. SigmaJS – JavaScript graphics library for drawing networks of connected nodes.
3. Plotly – High-level JavaScript graphing library; Used to create scientific plots and charts; Includes many tools for data viewing and manipulation.

At the outset of development for this application, we chose PaperJS, as the library for drawing the visual network component of EPANET models. Our reason for choosing this library over others was its ability to draw our water distribution networks, no matter how complicated. After a good deal of proof-of-concept development we found that the implementation of PaperJS to meet our needs would require much more time and customization than we felt appropriate for this project. We then went through the same prototyping process as before with SigmaJS. It proved to be much easier to implement and included key features that we needed out of the box. Some of these features were the ability to link nodes and edges together seamlessly, on click event handlers for nodes and edges, and the ability to customize properties of the network on the fly (such as colour of nodes or position of the camera).

4.3. Implementation

Before any of the technologies listed about could be implemented, we first needed a way to parse the EPANET model files, which are structured text files, into a format that that could be used in the code. The EPANETTOOLS documentation does not provide any specification that can be reliably used for this task. As a result, it became necessary to build a custom parsing script. The script closely resembles a tokenizer (Figure 7) that creates a JS object with keys at the root level representing each main section in a .inp file (EPANET models’ preferred extension). Although this presented more work than if the Python wrapper had had a function that met our needs, because we wrote a custom script, we could parse the files into a structure that we could pass directly into the initialization of a SigmaJS network graph. This made implementing SigmaJS significantly easier.

Given an EPANET .inp file parser, it then became necessary to implement a .inp file writer. After a model is edited and updated, the user would need to be able to execute the updated version through EPANETTOOLS model simulator. This requires an updated .inp file, thus the need for a .inp file writer. Unlike the parsing script’s tokenizer optimization, we could not identify an elegant way to use a JS object representation of an EPANET model and write it back into the structured text file that the EPANET model program expects. The writer follows a brute force approach and consists mostly of structured string concatenations with the right spacing and section separators (colons, semi-colons, etc.). The reader and writer scripts are another aspect of the EPANET Model Repository that is completely specific to EPANET models and therefore would provide little to no reproducible code for other communities looking to create similar web ecosystems.

The next step in implementation was creating a viewing window for the pipe network’s visual representation (Figure 8). As mentioned above, we created our model parsing script such that we could pass the objects created for nodes and links (nodes and edges in SigmaJS terminology) directly into the SigmaJS graph constructor. The ‘model’ object, that we pass into the graph setting of the Sigma object, consists of an array of nodes and an array of pipes. Each node in

```javascript
const inpType = {
  TITLE: "[TITLE]",
  JUNCTIONS: "[JUNCTIONS]",
  RESERVOIRS: "[RESERVOIRS]",
  TANKS: "[TANKS]",
  PIPES: "[PIPES]",
  PUMPS: "[PUMPS]",
  VALVES: "[VALVES]",
  TAGS: "[TAGS]",
  DEMANDS: "[DEMands]",
  STATUS: "[STATUS]",
  PATTERNS: "[PATTERNS]",
  CURVES: "[CURVES]",
  CONTROLS: "[CONTROLS]",
  RULES: "[RULES]",
  ENERGY: "[ENERGY]",
  EMITTERS: "[EMITTERS]",
  QUALITY: "[QUALITY]",
  SOURCES: "[SOURCES]",
  REACTION1: "[REACTIONS1]",
  REACTION2: "[REACTIONS2]",
  MIXING: "[MIXING]",
  TIMES: "[TIMES]",
  REPORT: "[REPORT]",
  OPTIONS: "[OPTIONS]",
  COORDINATES: "[COORDINATES]",
  VERTICES: "[VERTICES]",
  LABELS: "[LABELS]",
  BACKDROP: "[BACKDROP]",
  END: "[END]"
};
```

Figure 7. EPANET model reader script’s token definitions.
Figure 8. Query tool, nodes that meet criteria highlighted in blue.

The array is comprised of EPANET specific settings for its type and an X, Y coordinate. SigmaJS uses the coordinates to plot the points on the canvas (passed in as the container argument below). Each pipe in the array is comprised of EPANET specific settings for its type as well as a target and source node. SigmaJS graphically connects the pipe or edge between the target and source nodes. The last step in implementing SigmaJS was setting up the graph event listeners, including:

- ‘clickStage’ events include events like panning the view and adding new nodes.
- ‘clickNodes’ events include dragging node positions or opening the node settings/results modal.
- ‘clickEdges’ events include opening the pipe settings/results modal or selecting the pipe to remove it.

At this point in development we determined to allow the user to edit any of the model settings in an easy intuitive way. This included recreating the secondary graphical editor windows such as the Pattern Editor and Curve Editor windows. The graphical editor windows are comprised of standard HTML inputs and some manipulation of a PlotlyJS graph. For example, the Pattern Editor (Figure 12), was created with a line plot that take the multipliers given and artificially creates two points on the line, one for the beginning of the timestep and one for the end. This gives the continuous line plot that resembles what is happening over time for each pattern.

After the applications features included editing any setting in the EPANET input file we added features to analyse results of a model simulation including:

- A query tool that highlights nodes or pipes that meet the criteria identified in the tool’s inputs (Figure 8).
- A results page for individual nodes and pipes that opens when a user clicks on one in the viewing window (Figure 9).
- Time series results animation over entire pipe network (Figure 9).
- Combined results plot for nodes and pipes in network (Figure 11; Can display all nodes or edges at the same time; Legacy GUI only allows for 5 nodes or pipes at one time).

The time series animation was accomplished by first creating a continuous mapping of a colour scale to the range of values of a given result parameter (head, demand, pressure, flow, etc.). At each time step in the animation the colour property of each SigmaJS node and edge was set to its new value. A standard ‘setTimeout’ JS function, changing the timeout based on the indicated animation speed, creates the animation effect. The results graphs were all made with PlotlyJS. The creation of these plots is very simple. A user passes in the datapoints and define certain styling and the library does the rest of the work including zooming, panning and downloading the plot as a PNG image.
4.4. Model viewer results

Our goal for the EPANET Model Repository was to create a lightweight web-based GUI that closely mimicked the legacy desktop GUI’s functionality while improving on some of its UI/UX. This project proved to us how powerful existing opensource cyberinfrastructure tools are. We were able to create a powerful modelling tool that only requires an internet connection to utilize. We believe the EPANET Model Repository is aesthetically appealing while also providing valuable data analysis tools for EPANET models. The figures compare the EPANET legacy GUI for Windows and the EPANET Model Viewer created for this project. Significant differences are explained for each comparison. The same model will be used as a use case for both applications.

4.4.1. Main viewing window

The main view of each application shows the graphical representation of the pipe network (Figure 10). Each view below is showing the same time-step animation of simulation results. As shown, the tools on the updated application are larger and more accessible with more vibrant colours to discern values. It is notable that the Legacy GUI’s animation colour scale mapping isn’t a linear mapping. For example, the head values for this time step range from 125 to 307, not 0 to 100. This causes a slightly misleading initial analysis of the graph. The colour

Figure 9. Individual node results pages showing demand over extended period simulation.

Figure 10. Legacy GUI main view showing animation of model results (a) and EPANET model viewer main view showing animation of model results (b).
scale in the web based system is superior for this reason.

4.4.2. Simulation node results graph
The two plots are very similar timeseries representations of the data (Figure 11). The EPANET Model Viewer also provides a stats summary page that provides even more useful context to the analysis of a simulation. The generation of one of these plots from a user’s standpoint is more clunky with the legacy GUI as you have to manually find and add elements to the plot, whereas with the app we created, all of the elements that can be added to the graph are listed in one spot and are added to or removed from the plot at the click of a button. The largest difference between the two version however, is the Legacy app only allows for 5 elements to be plotted on the graph. Being able to plot as many of the elements in the pipe network as desired, at one time, provides extra insight into the bigger picture of what is happening in the simulated network.

4.4.3. Pattern editor
Both pattern editors (Figure 12) are very similar in the way one adds and views the pattern. This is one of the areas where the legacy GUI may be more functional, although more complex. The legacy GUI allows one to load or save existing patterns in the form of a CSV. This is something we may consider to add in the future.

4.4.4. Curve editors
Similarly, the curve editors (Figure 13) are very similar in form and function, however the Legacy GUI once again allows the user to load and save a curve in the form of a CSV file, functionality not supported by the Model Viewer curve editor.

4.4.5. Control editor
Controls are a more advanced model setting within an EPANET input file. One can control whether pipes are open or closed based on surrounding conditions of the pipe network or based on a time. The same
method of control exists for pump activation and reservoir outflows. In the legacy GUI adding a control is completely up to the user’s knowledge of the control syntax. This can lead to input file errors and added learning curves for using EPANET. The Control Editor we created knows the proper syntax and helps the user create controls by offering the options for each control type and parameter in the form of dropdown inputs. Having a more intuitive and simpler UI will hopefully allow more people to take advantage of this powerful setting for EPANET model simulation with the use of our app (Figure 14).

5. Discussion and conclusions

This experimental system demonstrated using existing open source building blocks to easily create powerful open web modelling applications using a number of web programming libraries and integration with HydroShare and Tethys Platform. Compared to the legacy GUI, our EPANET Model Repository and Model Viewer applications only require a modern browser and an internet connection to run. They are also both primarily comprised of Python, and JavaScript code, some of the most universal languages in use today, thus lowering the barriers for creating plugins and extensions.

Using the tools described above HydroShare and Tethys Platform, creating the EPANET model repository required writing less than 700 lines of code, which for a comprehensive web application is very good. A lot of the code the repository is comprised of is reusable, meaning, to adopt this structure to another modelling community would require far less lines of code, increasing the speed of development and lowering the learning curve even more.

The EPANET model viewer required significantly more lines of code, most of which being non-reusable due to the individual nature of a modelling system. However,

Figure 13. Legacy GUI curve editor (a) and EPANET model viewer curve editor (b).

Figure 14. Legacy GUI controls editor (a) and EPANET model viewer controls editor (b).
replicating nearly 100% of the legacy EPANET desktop GUI’s functionality into a light-weight web application, that anyone with a network connection can use, is a significant and powerful addition to the EPANET community. Incorporating model instances from HydroShare into the viewer app is most likely the only portion of the code that is directly usable by other modelling communities’ systems. Some of the graphics and drawing libraries with their respective implementations would most likely also be of use to other modelling systems that include a visual aspect to their editing and analysing.

We believe that the future of modelling and data science lies in the cloud. Giving anyone, anywhere, with an internet connection, access to powerful modelling tools and a vibrant community of modelling experts will provide more of the world with the benefits of these sciences. It will also increase the rate at which these sciences evolve and progress. The web ecosystem of modelling tools that we have created for this project are a step towards the future of modelling in the cloud. The workflow we have developed and presented here is quite simple and can be easily adapted by other modelling communities. After minimal changes, the EPANET Model Repository could be a catalogue for any type of model instance and can be tied into other web-based applications that are compatible with those models. Although the EPANET Model Viewer is a more specialized application, it too has a simple structure that can be adapted to other model types using the repository as its main datastore.

With the advent of the popularization of open source technologies, the barriers to creating web modelling applications using legacy models have been lowered dramatically. This project demonstrates that, with some time spent on creating a solid design plan and finding the right open source technologies, bringing existing modelling tools to the new age of cloud computing is not the daunting task it once was. Much like the impact that these cloud ecosystems will have on their individual scientific communities, as more modelling tools become available on the cloud other communities will take notice and follow suit. We hope that the structure we have presented in the project will play, at least a small roll, in the future of cloud-based open modelling.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

This work was supported by the US Environmental Protection Agency [RB835950].

**ORCID**

Daniel P. Ames [http://orcid.org/0000-0003-2606-2579](http://orcid.org/0000-0003-2606-2579)

**References**

Barton, C. M., M. Alberti, D. Ames, J.-A. Atkinson, J. Bales, E. Burke, M. Chen, et al. 2020. “Call for Transparency of COVID-19 Models.” *Science* 368: 482–483.

Burger, G., R. Sitzenfrei, M. Kleidporfer, and W. Rauch. 2016. “Quest for a New Solver for EPANET 2.” *Journal of Water Resources Planning and Management* 142: 04015065. doi:10.1061/(ASCE)WR.1943-5452.0000596.

Chen, M., A. Voinov, D. P. Ames, A. J. Kettner, J. L. Goodall, A. J. Jakeman, M. C. Barton, et al. 2020. “Position Paper: Open Web-distributed Integrated Geographic Modelling and Simulation to Enable Broader Participation and Applications.” *Earth-Science Reviews* 207: 103223. doi:10.1016/j.earscirev.2020.103223.

Crawley, S., D. Ames, Z. Li, and D. Tarboton. 2017. “HydroShare GIS: Visualizing Spatial Data in the Cloud.” *Open Water Journal* 4(1): 4–22.

Evans, S. W., N. L. Jones, G. P. Williams, D. P. Ames, and E. J. Nelson. 2020. “Groundwater Level Mapping Tool: An Open Source Web Application for Assessing Groundwater Sustainability.” *Environmental Modelling & Software* 131: 104782. doi:10.1016/j.envsoft.2020.104782.

Gan, T., D. G. Tarboton, P. Dash, T. Z. Gichamo, and J. S. Horsburgh. 2020. “Integrating Hydrologic Modeling Web Services with Online Data Sharing to Prepare, Store, and Execute Hydrologic Models.” *Environmental Modelling & Software* 130: 104731. doi:10.1016/j.envsoft.2020.104731.

Jiang, P., M. Elag, P. Kumar, S. D. Peckham, L. Marini, and L. Rui. 2017. “A Service-oriented Architecture for Coupling Web Service Models Using the Basic Model Interface (BMI).” *Environmental Modelling & Software* 92: 107–118. doi:10.1016/j.envsoft.2017.01.021.

Kadlec, J., A. W. Miller, and D. P. Ames. 2016. “Extracting Snow Cover Time Series Data from Open Access Web Mapping Tile Services.” *JAWRA Journal of the American Water Resources Association* 52 (4): 916–932. doi:10.1111/1752-1688.12387.

Khattar, R., M. M. Khattar, D. Ames, E. J. Nelson, and N. L. Jones. 2018. “Introducing the HydroShare App Nursery-A Sandbox for Testing, Developing, and Deploying GIS-Enabled Environmental Web Apps.” In *Proceedings of the 9th International Congress on Environmental Modelling and Software*, Fort Collins, Colorado, USA.

Menapace, A., D. Avesani, M. Righetti, A. Bellin, and G. Pisuturo. 2018. “Uniformly Distributed Demand EPANET Extension.” *Water Resources Management* 32: 2165–2180. doi:10.1007/s11269-018-1924-6.

Nativi, S., P. Mazzetti, and G. N. Geller. 2013. “Environmental Model Access and Interoperability: The GEO Model Web Initiative.” *Environmental Modelling & Software* 39: 214–228. doi:10.1016/j.envsoft.2012.03.007.

Overeem, I., M. M. Berlin, and J. P. M. Syvitski. 2013. “Strategies for Integrated Modeling: The Community Surface Dynamics Modeling System Example.” *Environmental Modelling & Software* 39: 314–321. doi:10.1016/j.envsoft.2012.01.012.

Peckham, S. D., E. W. Hutton, and B. Norris. 2013. “A Component-based Approach to Integrated Modeling in
the Geosciences: The Design of CSDMS.” *Computers & Geosciences* 53: 3–12. doi:10.1016/j.cageo.2012.04.002.

Perez, J. F., N. R. Swain, H. G. Dolder, S. D. Christensen, A. D. Snow, E. J. Nelson, and N. L. Jones. 2016. “From Global to Local: Providing Actionable Flood Forecast Information in a Cloud-Based Computing Environment.” *JAWRA Journal of the American Water Resources Association* 52: 965–978. doi:10.1111/1752-1688.12392.

Perkel, J. 2016. “Democratic Databases: Science on GitHub.” *Nature News* 538: 127. doi:10.1038/538127a.

Sadler, J. M., D. P. Ames, and S. J. Livingston. 2016. “Extending HydroShare to Enable Hydrologic Time Series Data as Social Media.” *Journal of Hydroinformatics* 18: 198–209. doi:10.2166/ hydro.2015.331.

Sayyed, M. A. H. A., R. Gupta, and T. T. Tanyimboh. 2015. “Noniterative Application of EPANET for Pressure Dependent Modelling of Water Distribution Systems.” *Water Resources Management* 29: 3227–3242. doi:10.1007/s11269-015-0992-0.

Singh, J. 2011. “FigShare.” *Journal of Pharmacology and Pharmacotherapeutics* 2: 138. doi:10.4103/0976-500X.81919.

Soranno, P. A., K. S. Cheruvelli, K. C. Elliott, and G. M. Montgomery. 2015. “It’s Good to Share: Why Environmental Scientists’ Ethics are Out of Date.” *BioScience* 65: 69–73. doi:10.1093/biosci/biu169.

Steffelbauer, D., and D. Fuchs-Hanusch. 2015. “OOPNET: An Object-oriented EPANET in Python.” *Procedia Engineering* 119: 710–718. doi:10.1016/j.proeng.2015.08.924.

Swain, N. R., S. D. Christensen, A. D. Snow, H. Dolder, G. Espinoza-Dávalos, E. Goharian, N. L. Jones, E. J. Nelson, D. P. Ames, and S. J. Burian. 2016. “A New Open Source Platform for Lowering the Barrier for Environmental Web App Development.” *Environmental Modelling & Software* 85: 11–26. doi:10.1016/j.envsoft.2016.08.003.

Tarboto, D. G., R. Idaszak, J. S. Horsburgh, J. Heard, D. Ames, J. L. Goodall, L. Band, et al. 2014. “HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing.” In *Proceedings of the 7th International Congress on Environmental Modelling and Software*, San Diego, California, USA.

Voinov, A., and F. Bousquet. 2010. “Modelling with Stakeholders.” *Environmental Modelling & Software* 25: 1268–1281. doi:10.1016/j.envsoft.2010.03.007.

Zhang, F., M. Chen, D. P. Ames, C. Shen, S. Yue, Y. Wen, and G. Lü. 2019. “Design and Development of a Service-oriented Wrapper System for Sharing and Reusing Distributed Geoanalysis Models on the Web.” *Environmental Modelling & Software* 111: 498–509. doi:10.1016/j.envsoft.2018.11.002.

Zhu, Y., and J. Yang. 2019. “Automatic Data Matching for Geospatial Models: A New Paradigm for Geospatial Data and Models Sharing.” *Annals of GIS* 25: 283–298. doi:10.1080/19475683.2019.1670735.