Journal of Hydroinformatics

Journal of Hydroinformatics Vol 26 No 2, 494 doi: 10.2166/hydro.2024.212

Client-side web-based model coupling using basic model interface for hydrology and water resources

Gregory Ewing [a,b,*], Carlos Erazo Ramirez [a,b], Ashani Vaidya[c] and Ibrahim Demir [a,b]

[a] IIHR – Hydroscience & Engineering, University of Iowa, Iowa City, IA, USA
[b] Civil and Environmental Engineering, University of Iowa, Iowa City, IA, USA
[c] Electrical and Computer Engineering, University of Iowa, Iowa City, IA, USA
*Corresponding author. E-mail: gregory-ewing@uiowa.edu

GE, 0000-0002-0106-7712; CER, 0000-0003-4337-2325; ID, 0000-0002-0461-1242

ABSTRACT

A recent trend in hydroinformatics has been the growing number of data, models, and cyber tools, which are web accessible, each aiming to improve common research tasks in hydrology through web technologies. Coupling web-based models and tools holds great promise for an integrated environment that can facilitate community participation, collaboration, and scientific replication. There are many examples of server-side, hydroinformatics resource coupling, where a common standard serves as an interface. Yet, there are few, if any, examples of client-side resource coupling, particularly cases where a common specification is employed. Toward this end, we implemented the basic model interface (BMI) specification in the JavaScript programing language, the most widely used programing language on the web. By using BMI, we coupled two client-side hydrological applications (HydroLang and HLM-Web) to perform rainfall–runoff simulations of historical events with rainfall data and a client-side hydrological model as a case study demonstration. Through this process, we present how a common and often tedious task – the coupling of two independent web resources – can be made easier through the adoption of a common standard. Furthermore, applying the standard has facilitated a step toward the possibility of client-side 'Model as a Service' for hydrological models.

Key words: basic model interface, hydroinformatics, integrated modelling, web-based simulation, web frameworks

HIGHLIGHTS

- We present the basic model interface (BMI) specification for the JavaScript programming language.
- We present a comprehensive example of how BMI may be used as a common standard to couple client-side, hydroinformatics web resources.
- For developers, BMI for JavaScript simplifies the effort needed to implement an Application Programing Interface for their resource.
- For users, BMI for JavaScript accelerates learning and working with a new resource.

1. INTRODUCTION

A recent trend in hydroinformatics has been the growing number of data, models, and other tools which are web accessible. For example, online community services are openly available, which allow people to connect to and leverage server-side resources to perform modeling, simulation, and data storage tasks, e.g., Community Surface Dynamics Modelling System (CSDMS) (Peckham 2015) and HydroShare (Horsburgh *et al.* 2016). Likewise, governmental and proprietary services offer access to environmental and hydrometeorological data through interactive web services and Application Programing Interfaces (APIs). Individually, these products each aim to improve common tasks and subtasks in hydrology with web technologies. Together, these open web services and community-driven repositories hold great promise to create a web-based environment that meets the community goals of access, participation, documentation, distribution, and scientific replication.

There are many efforts for web-based data analytics (Xu *et al.* 2019; Alabbad *et al.* 2022), information portals (Sit *et al.* 2021a; Yildirim & Demir 2021), and decision systems (Ewing & Demir 2021) in hydrology and water resources. In comparison, community-focused modeling frameworks and integration is rather limited. Coupling hydrological models with web-

accessible data is presently a nontrivial task. Resource coupling can require a large amount of programing time and effort, specific understanding of each product's codebase (and their idiosyncrasies), and, typically, input from source code authors (Peckham 2014). Issues may arise when two or more distinct models are coupled (e.g., having to merge and recreate features and methodologies), ultimately limiting the scalability and reproducibility required for community growth. These activities put a significant burden on end users, limit the extent, and influence the speed at which hydrologic data, models, and applications can integrate for beneficial use for the research and the public. Thus, there still exists a need to further lower barriers to couple web-enabled, hydrologic resources.

One option to standardize the coupling of modeling and data resources is the basic model interface (BMI). BMI is a specification library containing a set of standard control and query functions with predefined inputs and outputs, along with data type descriptions and any other information pertaining to specific models (Hutton *et al.* 2020). BMI specification promotes loosely coupled models and data resources that can be reused and scaled by the community without internally manipulating resources. Currently, the BMI specification is available in five programing languages: C, C++, Fortran, Java, and Python (CSDMS 2022).

BMI has been applied to web-based hydrological applications – e.g., Tethys Platform (Swain *et al.* 2015) and the Group on Earth Observations Platform (Giuliani *et al.* 2013) – and workflow systems – e.g., the MINT framework (Gil *et al.* 2018) and the 3Di platform (Baart *et al.* 2014; de Boer *et al.* 2014). Further, CSDMS has applied BMI to a web service architecture that allows for fast computing and data resources (Jiang *et al.* 2017).

To date, the use of BMI on the web has primarily focused on its specification and standard naming conventions to facilitate directives from the client to a server-based resource. Thus, many of these applications rely exclusively on using server-side resources for running models, and clients' local hardware for browser-based interfaces (e.g., make calls to the servers, receive, and display data). Thus, there are still opportunities to further couple services, particularly those operating in client-side environments. In the literature, there are few, if any, examples of client-side hydroinformatics resource coupling. Moreover, no implementation of BMI has been found that relies entirely on JavaScript, the most widely used and suitable programing language for web development.

The previous work in server-side resource coupling provides a strong case to adopt a standard-based approach for client-side resources. Toward this end, we implemented the BMI specification in the JavaScript programing language. Later through the BMI specification, we coupled two client-side hydrological applications (HydroLang and HLM-Web) to perform rainfall–runoff simulations of historical events as a case study demonstration.

The remainder of this article is as follows. In Section 2, we outline the BMI implementation in JavaScript; describe the client-side resources used, HLM-Web and HydroLang, and the process to make them BMI compliant; and outline the catchment and the data sources used for the simulation. In Section 3, we share the procedure to couple the client-side resources to support rainfall–runoff modeling of a historical event. We also present the simulation results of a catchment using HLM-Web with different web-available precipitation products. Further, we discuss the use of the BMI specification in JavaScript and its ability to facilitate a web-based hydrological workflow. Finally, we conclude with potential areas for the future work.

2. MATERIALS AND METHODS

In this section, we first describe the implementation of the BMI specification in JavaScript. Next, we describe the client-side modeling and programing frameworks, HLM-Web, and HydroLang. Finally, we describe the study area and data used to simulate historical events for the case study.

2.1. BMI-JS

The BMI standard prescribes to developers a set of functions to implement on their model or data source, which enables it to communicate to external resources through syntactical variable naming. In total, the current version of the standard describes 41 functions, falling within six functional groups (Table 1), which must be implemented for a resource to be considered BMI compliant (Hutton *et al.* 2020).

The design pattern for BMI in JavaScript was chosen to be a class, where all functions detailed in the standard were included as methods in the BMI class. This pattern largely mimics the pattern used in the Python implementation of BMI due to its similarity as a high-level, object-orient language.

Table 1 | BMI categorical groups, adopted from CSDMS (2022)

Functional group	Examples
Model control	*initialize, update, finalize*
Model information	*get_input_var_names, get_output_item_count*
Variable information	*get_var_units, get_var_location*
Time	*get_start_time, get_time_step*
Variable getter and setter	*get_value, set_value*
Model grid	*get_grid_size, get_grid_shape*

Resources using BMI.js can implement all but one of the BMI's specifications, *get_value_ptr*; unlike other languages, it is not possible in JavaScript to pass references or pointers. Assumptions in the use of BMI.js include the user's understanding of the language, e.g., base JavaScript's limited primitive numerical data types, thread, and memory management.

We refer the reader to Supplemental Material for more information about the package, including a link to its public code repository.

2.2. HLM-Web

HLM-Web is a modeling framework and simulation engine capable of performing distributed, physically based rainfall–runoff simulations entirely in a modern web browser. Its development and uses have been reported in the study by Ewing *et al.* (2022). HLM-Web is the JavaScript implementation of the hillslope-link model (HLM) (Mantilla 2022).

HLM uses the concept of landscape decomposition into just two elements: hillslopes and channel links. Hillslopes are irregular-shaped polygons where the conversion of rainfall to runoff takes place. Each hillslope is uniquely associated with a channel link. A hillslope's runoff, both surface and subsurface, drains to the corresponding link and the links make the river network. Mathematically, the physical processes involved in runoff generation and transport are described as a set of ordinary differential equations. Within the HLM framework, there is a good deal of flexibility for the level of detail desired to describe physical processes. The simulations presented here use a nonlinear concept of storage at the hillslope level and a nonlinear channel velocity.

HLM-Web uses the object-oriented programing paradigm with the hillslope-link pair as the basis for the object model. Each hillslope-link object contains all necessary components to step forward the hillslope's simulation as object methods and properties. In general, these methods and properties can be categorized into four groups: forcing data (i.e., evaporation potential and precipitation), physical attributes (e.g., hillslope area, total upstream area, channel length), model structure (equations that describe physical processes), and numerical solver. All necessary properties and methods are added at object instantiation via input and configuration files and can also be altered prior to simulation execution.

2.2.1. BMI implementation

To comply with the BMI specification, the HLM-Web code required some refactoring. The original HLM-Web codebase uses the hillslope links as the basis for the object model. Yet, implementing BMI on each hillslope-link object would not be consistent with the intent of BMI because internal model elements (i.e., hillslope links) would have been exposed to linked resources, instead of the high-level model itself. Thus, to achieve BMI compliance *and* vision, hillslope-link objects are packaged, or wrapped, within a new HLM class. This new HLM class then complies with the BMI specification through inheritance from the BMI class and the HLM-specific implementation of its methods. The HLM class also holds the metadata associated with the simulation and other methods (not named in the BMI specification) that are integral to the orchestration of the numerical modeling and functionality of the BMI-specified methods. For example, the BMI-specified *initialized* method calls methods on the BMI-compliant HLM class to configure model objects and the numerical solver. These helper methods were added to HLM to achieve BMI compliance. Very little information has been lifted to the HLM class from the individual hillslope-link objects, in effect keeping their functionality the same. We refer the reader to Supplemental Material for more information about BMI-compliant HLM-Web, including a link to its public code repository.

2.3. Hydrolang

HydroLang.js is a community-driven, open-source computational framework for hydrological research and education (Erazo *et al.* 2022). HydroLang.js supports programming using both JavaScript and HTML like markup language (Erazo Ramirez *et al.* 2023). HydroLang.js consists of four low cohesive modules: (1) *Data*, for retrieving data from various data sources (i.e., APIs); (2) *Visualize*, for data visualization; (3) *Maps*, for creating maps rendered on the browser to view georeferenced data and data selection; and (4) *Analyze*, for performing hydrological, statistical, and neural network subroutines used in academia and industry. The library was created using JavaScript for its ease of online application development, integrating with other open-source libraries while maintaining industry standards, and having lexical commands that aid in implementation. HydroLang greatly decreases the amount of programing required to do any of the included subroutines, allowing users to complete analysis in just a few lines of code as a one-stop resource.

2.3.1. BMI implementation

When implementing BMI compliance into HydroLang, it was important to maintain the original framework's functionality. To achieve this, a 'HydroLangBMI' class was created to extend the framework, adding the BMI functions but only as method callers through configuration files. As a result, HydroLang BMI achieves compliance without the need to reprogram existing methods. A new instance of the compliant class is made through variable declaration (i.e., 'new HydroLangBMI('config.json')'). All HydroLang modules and functions can be accessed using configuration files with BMI implementation, provided they match the framework's criteria.

To access the functionalities of the framework for a specific use case or requirement, a new instance of the parent class (HydroLangBMI) is created. This new instance inherits methods from the BMI and the HydroLang wrapper, while also allowing users to add their own helper functions. This decision was made to enable users to customize the implementation of a BMI-compliant resource without having to recreate the entire implementation. It provides the flexibility to override the wrapper's functionality as needed.

For the case study described in the following section, a new class instance called HLData was created. This instance inherits from the HydroLangBMI class and includes additional helper functions that make API calls to different online resources. The preprocessed data are then accessed by the HLM-BMI rainfall–runoff model. We refer the reader to Supplemental Material for more information about HydroLangBMI, including a link to its public code repository.

2.4. Case study and experimental setup

Simulation and study of historical events is an important hydrological component of many research and engineering design studies. These studies require the collection and use of a myriad of data for both model construction and model forcing. There exists a clear delineation between a hydrological model and its forcing data, which provides a meaningful opportunity to demonstrate the potential for loosely coupled resources to perform a collective task. Thus, we chose to demonstrate the coupled capabilities of HLM-Web and HydroLang via BMI by simulating a historical rainfall event of a HUC-10 watershed, the Clear Creek watershed (HUC-10: 0708020904). For more information relating to the Clear Creek watershed and its use in this case study, please refer to the Supplemental Material.

Two rainfall products were used for this case study. The first data product is preprocessed Multi-Radar Multi-Sensor (MRMS), data retrieved from and hosted by the Iowa Flood Center for flood forecasting in Iowa (Smith *et al.* 2016; Quintero *et al.* 2020). These data were transformed into JSON (RFC 2014) and hosted as a public dataset accessible via an API to communicate with the HydroLang data service. The second data product is a gridded, hourly precipitation model at 0.125 resolution from the North American Data Assimilation System (NLDAS) (Mitchell *et al.* 2004). Data are accessed externally via the Environmental Protection Agency's (EPA), Hydrologic Micro Services (Parmar *et al.* 2018), which provides historical data from different rainfall products.

For each data model, an instance of the HydroLangBMI class was used to retrieve, analyze, and change the service calls' retrieved data. With the MRMS product, data requests are accessible synchronously as the configuration file is loaded into the model. With the NLDAS product, data requests are asynchronous. In all situations, the call results were internally modified and stored in a global variable to await BMI-compliant function calls.

3. RESULTS AND DISCUSSION

In this section, we share the procedure to couple the client-side resources to support rainfall–runoff modeling of historical events. We also present the simulation results of a study catchment using HLM-Web with different web-available

precipitation products. Further, we discuss the use of the BMI specification in JavaScript and its ability to facilitate a web-based, hydrological modeling workflow.

Shared getter and setter methods couple BMI-compliant versions of HLM and HydroLang Figure 1. Our approach first initializes each resource and then performs QA/QC checks to ensure compatibility. Next, a loop advances the simulation. Within the loop, HydroLang first retrieves precipitation information at model time, and then HLM-Web simulation uses these data as precipitation. Finally, each resource is updated after the values are set. After the simulation, each resource is shut down and HLM-Web simulation results are graphed with the HydroLang visualization module.

Simulations of the rainfall–runoff response of Clear Creek watershed were performed using both preprocessed MRMS precipitation data and data gathered directly from an EPA-hosted, grid-based rainfall API for a 17-day period, from 28 September 2018 to 15 October 2018. All simulations were performed in Google's Chrome browser, with the V8 JavaScript interpreter, on a Dell XPS 9560 with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz. The results demonstrate the ability of HLM-Web to successfully capture discharge pulses from the Clear Creek watershed with both rainfall products, for this historical event (Figure 2). However, general conclusions regarding model performance or suitability are beyond the scope of this study. The performance of a distributed flood forecasting model, such as HLM, is not determined by a single event for a single location within the model domain. For an investigation of HLM performance, we refer readers to the literature (Velásquez *et al.* 2021). Likewise, we refer the reader to Supplemental Material, where they can find resources to code used to perform the coupled simulation.

3.1. Discussions

Increasingly, JavaScript is used in both client-side and server-side environments. This versatility offers a single language for use at any level of the development stack. Owing to this versatility, JavaScript continuously ranks as one of the most used programing languages over the past decade (StackOverflow 2021). Further, JavaScript possesses many characteristics that make it a candidate for use in a number of science-related applications (Walker & Chapra 2014).

The BMI standard facilitates model development and coupling, as described in the literature (Goodall & Peckam 2016; Jiang *et al.* 2017). For example, the standard's agnostic framework enables it to be applied to any model and allows the flexibility of the developer to apply the internal design patterns that they see fit. This feature is demonstrated in the differing

```
Require: config-hlm-web.json, config-hydrolang.json
Ensure: Two models stepped to end of simulation time
  HLModel ← new instance of HLM from config-hlm-web.json
  dataModel ← new instance of HydroLang from config-hydrolang.json
  errors ← Check for startup errors
  if errors then
      break, throw error;
  end if
  while simulation time < simulation end do
      \\ use method: get_values_at_indices
      precipValues ← current precipitation values from dataModel

      \\ use method: set_values_at_indices
      HLM precip values at each link ← precipValues

      \\ use method: update
      update HLModel one time step
      update dataModel one time step
  end while
  \\ use method: finalize
  tear down models, output simulation results
```

Figure 1 | Pseudocode to couple the client-side data and model resources via the BMI specification.

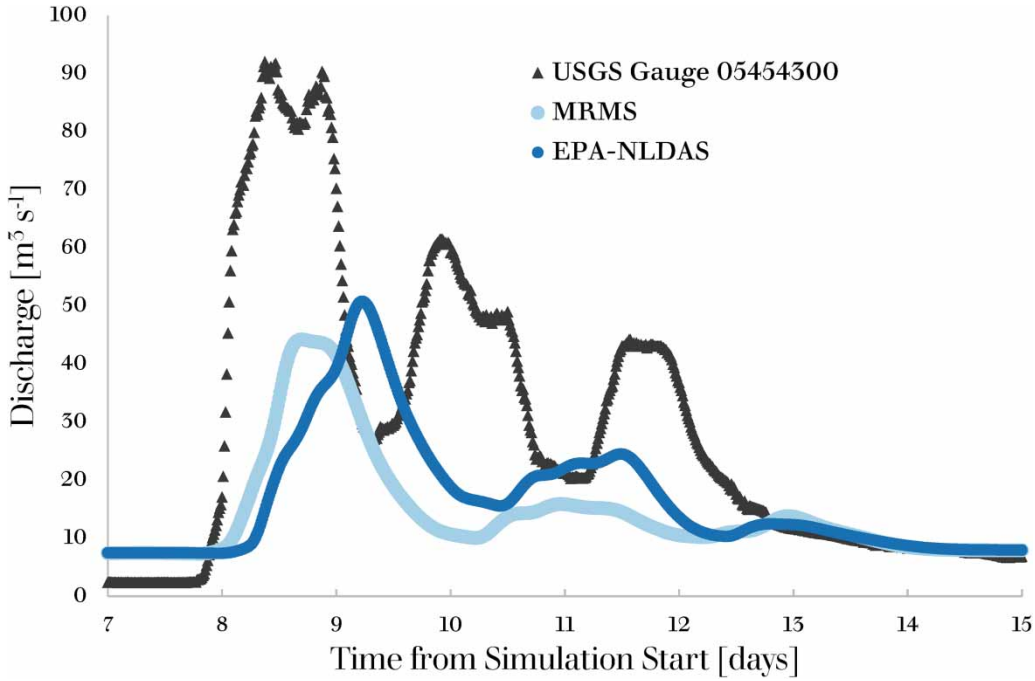

Figure 2 | Rainfall–runoff results of HLM-Web using precipitation forcing from MRMS and NLDAS. Also included are measurements at USGS Gauge 05454300. Simulations were performed on the interval September 28, 2018 to October 15, 2018. Captured here is the response to the 5 October 2018 event until simulation ends.

approaches that were taken to make both HydroLang and HLM-Web BMI compliant. Further, the standard allows easy inter-model communication via its use of standard naming for both the specified functions (i.e., object methods) and the standard naming for model variables. This feature is demonstrated via the application of BMI to model historical rainfall events. These features can benefit a wide range of modeling domains.

Upon reflection, we observe the benefit of a standard-based approach for resource coupling in a web environment. First, the use of a common standard such as BMI as the template for an API simplifies the effort needed to implement a functional API for most projects. We make this claim based on our experiences coupling HLM-Web and HydroLang via the BMI standard. Importantly, we consider this effort in comparison to the counterfactual, where the authors of each resource (HLM-Web and HydroLang, or any other resource for that matter) independently undertake a whole-cloth effort of designing and implementing an API from scratch (i.e., designing data flows, data structures, input and output, methods, etc.) *prior to the act of implementation of the API*. This difference in focus of effort can represent significant time savings.

Second, we assert that adopting a common API advantages users. A common API *should* accelerate learning and working with a new resource. Again, consider the counterfactual, where all resources have different design patterns and API implementations; the level of effort to use would increase with each new resource a user desires to interact with. Users suffer a much lower penalty when resource coupling with a common API. Likewise, as new resources become available and coupled through a common standard, the effort required by other users to do the same decreases. This is not always the case when 'one–off' connections are made.

Indeed, both HLM-Web and HydroLang benefited from adding BMI compliance. For HLM-Web, the refactored version encapsulates all model elements within a single model object. This design feature creates a clear scope to model elements and standardizes how users manipulate the model and interact with its features. Likewise, the BMI refactoring encouraged an improved model setup. For example, all input files now have common top-level keys, and the refactored code handles the input files in a similar manner. This makes model input data more descriptive and affords more flexibility of input.

Furthermore, refactoring HLM-Web to be BMI compliant can be exploited to improve memory management and performance of the modeling resource. This means that the BMI-compliant HLM-Web can perform simulations with longer time periods and models with larger physical extents (i.e., larger catchments) compared to its non-BMI counterpart.

This is achieved by employing two BMI-related improvements. First, large datasets of long-term time series forcings no longer need to be held in memory in the HLM model object itself, but rather in a BMI-compliant data model and subsequently queried at the desired timestep. Prior, all timeseries data would be ingested at once at model setup with other input data. Thus, for large precipitation datasets, the performance would be limited by the heavy memory usage and search time required to supply forcing data to the model elements during simulation. Second, by exploiting BMI's model control functions, e.g., *update_until*, memory use can be lowered by writing results externally at a greater frequency, subsequently requiring fewer intermediate results to be stored continuously in an HLM model instance.

For HydroLang, BMI compliance improves its flexibility, and scalability as an alternative interface. First, configuration objects are flexible and easy to use. Instead of invoking methods or functions, the 'initialize' function of the BMI standard allows the framework's usage through configuration objects that contain user-prescribed methods. Second, configuration objects enable easy data query changes. This feature allows data service reuse by adjusting location, timeframe, or queried variables. Finally, users can access several functions from different modules in a single BMI class instance. For example, visualization or data analysis elements can be added to a class instance and be used with external or simulated data.

From an operational perspective, coupling via BMI made the simulation workflow easier. Using the data service to supply forcing data meant that only data required for a given simulation was loaded locally. This removed the overhead of managing a database or local data store directly.

Limitations, however, do exist to both the applicability of client-side technologies to scientific computation and the standard-based approach to couple client-side resources. As compared to the other languages that are heavily used for modeling, there are clear limitations on JavaScript's performance. Many of these limitations stem directly from its interpretative nature and lack of low-level control. First, runtimes of scripts with heavy computation are slowed due to sluggish bitwise operations, which convert 32-bit signed integers to 64-bit floating point numbers. For example, a recent like-for-like comparison between JavaScript and C++ (i.e., a compiled language) suggests that JavaScript is approximately two times slower (Hinkelmann 2019). Second, web browsers were originally designed with a single thread to interpret and perform JavaScript code. This design limits multithreading and multiprocessing, where tasks can be efficiently lifted by the end user's environment. Recently, however, the web standard WebWorkers offers some workarounds to multithreading (W3C 2021). Finally, JavaScript does not support multiple inheritance, a feature widely used when developing environmental models. There is, however, an object-oriented workaround to achieve similar functionality.

There are also limits to the benefit of adopting a web-based, connected simulation workflow. For example, though HydroLang allows for easy integration of external data sources via APIs, it cannot guarantee its data quality, which is a common limitation for any programing library. The number of calls available to retrieve the required data is limited by the type of resource used to serve the data (a model or real-time acquired data) and the number of calls available to serve the data (request limits). For the development of this project, the NLDAS data quality process via automated quality control (Xia *et al.* 2019) ensures that the data obtained from the model are continuously monitored. Similarly, the preprocessed MRMS data are backed up by the team in charge of quality control and assurance (Tang *et al.* 2020). In short, easily connecting to a data resource does not dismiss the due diligence required before its use.

Although there are limitations, there remain strong benefits to simulation and modeling in JavaScript, as evidenced by the ease with which external data and visualization packages can be employed. Web-based environments commonly utilize JSON files for various purposes, such as configuration objects for package management and data retrieval/manipulation. The extensive adoption of this widely recognized standard significantly enhances the extensibility of BMI models.

4. CONCLUSION

The BMI is a specification library containing a set of standard control and query functions. Previously, BMI has been implemented in the programing languages commonly used for computational modeling in geosciences. Further, the previous work in the literature has demonstrated how the qualities of the BMI specification can be leveraged to achieve model/resource coupling in offline and server-side environments. Here, we first present the implementation of the specification for the JavaScript programing language. We then use the BMI specification to couple web-based data resources via a hydrological programing library and a web-based rainfall–runoff model. Finally, we use these coupled, web-based resources to simulate the historical hydrological response of a study watershed to a rainfall event. Through this process, we show how a common and often tedious task – the coupling of two independent web resources – can be made easier through the adoption

of a common standard. Furthermore, applying the standard has facilitated a step toward the possibility of client-side model as a service (MaaS) (Dumitru *et al.* 2009) for hydrological models.

Future work related to BMI-compliant coupling on the web broadly takes on three overlapping fronts. The first is to implement the specification for other web technologies. For example, WebGL (Khronos 2022) is used to quickly render 2d and 3d visualizations in browsers for web-based communication and virtual reality applications in hydrology (Sermet & Demir 2020). Mathematical transformations can be applied natively within the standard, allowing model and simulation tasks to be performed on a computer's graphics hardware. Results could be accessed via the BMI standard and used elsewhere. Second, expanding the application of BMI to other resources would allow easy connectivity to a further range of data and models; for example, we propose adding BMI compliance to parsimonious flood extent estimation models (Hu & Demir 2021; Li *et al.* 2022) and data-driven forecast models (Sit *et al.* 2021b). The third front of future work is to develop and share fit-to-purpose web applications that rely on BMI-coupled resources. Such applications could be in-browser flood event forecasting, where users can easily mix and match model forcings (i.e., MRMS, NLDAS, etc.) and different BMI-compliant models (e.g., HLM-Web, data-driven machine learning models, National Water Model, etc.) Given the ease of coupling, many more applications abound and can be as diverse as the community of modelers and developers who use them.

DATA AVAILABILITY STATEMENT

All relevant data are available from an online repository or repositories, as detailed in the supplementary material.

CONFLICT OF INTEREST

The authors declare there is no conflict.

REFERENCES

Alabbad, Y., Yildirim, E. & Demir, I. 2022 Flood mitigation data analytics and decision support framework: Iowa Middle Cedar Watershed case study. *Science of the Total Environment* **814**, 152768.

Baart, F., Ha, J., van Dam, A., Donchyts, G. & Siemerink, M. 2014 Interactive web-based flood modeling at country wide scale and plantar size resolution. In *International Congress on Environmental Modelling and Software*, San Diego.

CSDMS 2022 *BMI Documentation, CSDMS*. Available form: https://bmi.readthedocs.io/en/latest/.

de Boer, G., Baart, F., Becker, P. J. & Jagers, B. 2014 Overview Of Coupling Of Data Models And Information Through The Web Using Existing Standards.' In *International Conference of Hydroinformatics*, edited by CUNY Academic Works. New York.

Dumitru, R., Schade, S., Berre, A., Bodsberg, N. & Langlois, J. 2009 Model as a Service (MaaS). In: *AGILE Workshop – Grid Technologies for Geospatial Applications*. Hannover.Erazo C., Sermet Y., Molkenthin F., & Demir I. 2022. HydroLang: An Open-Source Web-Based Programming Framework for Hydrological Sciences, *Environmental Modelling & Software*, 157, p.105525.

Erazo Ramirez, C., Sermet, Y., Molkenthin, F. & Demir, I. 2022 HydroLang: an open-source web-based programming framework for hydrological sciences. *Environmental Modelling & Software* **157**, 105525. https://doi.org/https://doi.org/10.1016/j.envsoft.2022.105525.

Erazo Ramirez, C., Sermet, Y. & Demir, I. 2023 Hydrolang markup language: Community-driven web components for hydrological analyses. *Journal of Hydroinformatics*. https://doi.org/10.2166/hydro.2023.149.

Ewing, G. & Demir, I. 2021 An ethical decision-making framework with serious gaming: A smart water case study on flooding. *Journal of Hydroinformatics* **23** (3), 466–482.

Ewing, G., Mantilla, R., Krajewski, W. & Demir, I. 2022 Interactive hydrological modelling and simulation on client-side web systems: An educational case study. *Journal of Hydroinformatics* **24** (6), 1194–1206. https://doi.org/10.31223/x5nw6j.

Gil, Y., Couburn, K., Deelman, E., Duffy, C. & Ferreira da Silva, R. 2018 MINT: Model INTegration through knowledge-powered data and process composition. In: *International Congress on Environmental Modelling and Software* (BYU, ed.). BYU, Ft. Collins, CO.

Giuliani, G., Ray, N., Schwarzer, S., De Bono, A., Peduzzi, P., Dao, H., Van Woerden, J., Witt, R., Beniston, M. & Lehmann, A. 2013 Sharing Environmental Data through GEOSS. In I. Management Association (Ed.), Geographic Information Systems: Concepts, Methodologies, Tools, and Applications (pp. 1260–1275). IGI Global. https://doi.org/10.4018/978-1-4666-2038-4.ch076.

Goodall, J. & Peckam, S. 2016 Interoperability between the Basic Modeling Interface (BMI) and the Open Modeling Interface (OpenMI): A step toward building the earth system bridge for modeling framework interoperability. In *International Congress on Environmental Modelling and Software*, Toulouse.

Hinkelmann, F. 2019 Speed, speed, speed: JavaScript vs C++ vs webassembly. In *International JavaScript Conference*, New York.

Horsburgh, J., Morsy, M., Castronova, A., Goodall, J., Gan, T., Yi, H., Stealey, M. & Tarboton, D. 2016 Hydroshare: Sharing diverse environmental data types and models as social objects with application to the hydrology domain. *Journal of the American Water Resources Association* **52**, 873–889.

Hu, A. & Demir, I. 2021 Real-time flood mapping on client-side web systems using hand model. *Hydrology* **8** (2), 65.

Hutton, E., Piper, M. & Tucker, G. 2020 The Basic Model Interface 2.0: A standard interface for coupling numerical models in the geosciences. *Journal of Open Source Software* **5**, 2317.

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

Khronos 2022 WebGL 2.0 Specification. Khronos Group. Available form: https://registry.khronos.org/webgl/specs/latest/2.0/.

Li, Z., Mount, J. & Demir, I. 2022 Accounting for uncertainty in real-time flood inundation mapping using HAND model: Iowa case study. *Natural Hazards* **112** (1), 977–1004.

Mantilla, R., Krajewski, W., Velásquez, N., Small, S., Ayalew, T., Quintero, F., Jadidoleslam, N. & Fonley, M. 2022 The Hydrological Hillslope-Link Model for Space-Time Prediction of Streamflow: Insights and Applications at the Iowa ßlood Center. In: *Extreme Weather Forecasting, edited by Marina Astitha and Efthymios Nikolopoulous* 1st ed., 200–238. Þlsevier. https://www.elsevier.com/books/extreme-weather-forecasting/astitha/978-0-12-820124-4.

Mitchell, K., Lohmann, D., Houser, P., Wood, E., Schaake, J., Robock, A., Cosgrove, B., Sheffield, J., Duan, Q., Luo, L., Higgins, R., Pinker, R., Tarpley, J., Lettenmaier, D., Marshall, C., Entin, J., Pan, M., Shi, W., Koren, V., Meng, J., Ramsay, B. & Bailey, A. 2004 The multi-institution North American Land Data Assimilation System (NLDAS): Utilizing multiple GCIP products and partners in a continental distributed hydrological modeling system. *Journal of Geophysical Research (Atmospheres)* **109**, D07S90.

Parmar, R., Knightes, C., Smith, D., Wolfe, K., Koblich, J., Sitterson, J., Johnston, J. M. & Purucker, T. 2018 Hydrologic Micro Services. In: *9th International Congress on Environmental Modelling and Software*, edited by BYU. Fort Collins.

Peckham, S. 2014 EMELI 1.0: An Experimental Smart Modeling Framework for Automatic Coupling of Self-Describing Models. In: *International Conference on Hydroinformatics*, edited by CUNY Academic Works. New York: ICH.

Peckham, S. 2015 *Integated Plug-and-Play Modeling: An Overview of CSDMS and the Earth System Bridge Project*. University of Colorado. Available from: https://watershed.ucdavis.edu/files/content/files/CSDMS_ESB_Overview.pdf.

Quintero, F., Krajewski, W., Seo, B. & Mantilla, R. 2020 Improvement and evaluation of the Iowa Flood Center Hillslope Link Model (HLM) by calibration-free approach. *Journal of Hydrology* **584**, 124686.

RFC 2014 The JavaScript Object Notation (JSON) Data Interchange Format Standard. RFC.

Sermet, Y. & Demir, I. 2020 Virtual and augmented reality applications for environmental science education and training. In: *New Perspectives on Virtual and Augmented Reality*, pp. 261–275.

Sit, M., Langel, R. J., Thompson, D., Cwiertny, D. & Demir, I. 2021a Web-based data analytics framework for well forecasting and groundwater quality. *Science of The Total Environment* **761**, 144121.

Sit, M., Demiray, B. & Demir, I. 2021b Short-term hourly streamflow prediction with graph convolutional gru networks. *arXiv preprint* arXiv:2107.07039.

Smith, M., Lakshmanan, V., Stumpf, G., Ortega, K., Hondl, K., Cooper, K., Calhoun, K., Kingfield, D., Manross, K., Toomey, R. & Brogden, J. 2016 Multi-Radar Multi-Sensor (MRMS) severe weather and aviation products: Initial operating capabilities. *Bulletin of the American Meteorological Society* **97**, 1617–1630.

StackOverflow 2021 *2021 Developer Survey*. Available form: https://insights.stackoverflow.com/survey/2021.

Swain, N., Latu, K., Christensen, S., Jones, N., Nelson, E., Ames, D. & Williams, G. 2015 A review of open source software solutions for developing water resources web applications. *Environmental Modelling & Software* **67**, 108–117.

Tang, L., Zhang, J., Simpson, M., Arthur, A., Grams, H., Wang, Y. & Langston, C. 2020 Updates on the radar data quality control in the MRMS quantitative precipitation estimation system. *Journal of Atmospheric and Oceanic Technology* **37**, 1521–1537.

Velásquez, N., Mantilla, R., Krajewski, W., Fonley, M. & Quintero, F. 2021 Improving hillslope link model performance from non-linear representation of natural and artificially drained subsurface flows. *Hydrology* **2021** (8), 187. https://doi.org/10.3390/hydrology8040187.

W3C 2021 *Web Workers API. W3C Web Worker Standard*.

Walker, J. & Chapra, S. 2014 A client-side web application for interactive environmental simulation modeling. *Environmental Modelling & Software* **55**, 49–60.

Xia, Y., Zengchao, H., Chunxiang, S., Yaohui, L., Meng, J., Tongren, X., Xinying, W. & Baoqing, Z. 2019 Regional and global land data assimilation systems: Innovations, challenges, and prospects. *Journal of Meteorological Research* **33**, 159–189.

Xu, H., Demir, I., Koylu, C. & Muste, M. 2019 A web-based geovisual analytics platform for identifying potential contributors to culvert sedimentation. *Science of the Total Environment* **692**, 806–817.

Yildirim, E. & Demir, I. 2021 An integrated flood risk assessment and mitigation framework: A case study for middle Cedar River Basin, Iowa, US. *International Journal of Disaster Risk Reduction* **56**, 102113.

First received 6 September 2023; accepted in revised form 22 December 2023. Available online 16 January 2024