Environmental Modelling and Software 108 (2018) 133–153

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Advancing the Open Modeling Interface (OpenMI) for integrated water resources modeling

Caleb A. Buahin*, Jeffery S. Horsburgh

Department of Civil and Environmental Engineering and Utah Water Research Laboratory, Utah State University, Logan, UT, USA

ARTICLE INFO

Keywords:
Component-based modeling
model coupling
OpenMI
Integrated water resources modeling
High performance heterogeneous computing

ABSTRACT

The use of existing component-based modeling frameworks for integrated water resources modeling is currently hampered for some important use cases because they lack support for commonly used, topology-aware, spatiotemporal data structures. Additionally, existing frameworks are often accompanied by large software stacks with steep learning curves. Others lack specifications for deploying them on high performance, heterogeneous computing (HPC) infrastructure. This puts their use beyond the reach of many water resources modelers. In this paper, we describe new advances in component-based modeling using a framework called HydroCouple. This framework largely adopts the Open Modeling Interface (OpenMI) 2.0 interface definitions but demonstrates important advances for water resources modeling. HydroCouple explicitly defines standard and widely used geospatial data formats and provides interface definitions to support simulations on HPC infrastructure. In this paper, we illustrate how these advances can be used to develop efficient model components through a coupled urban stormwater modeling exercise.

1. Introduction

The goal of integrated assessment in environmental and natural resources management is to provide information within a decision making context that brings together a broader set of domains, methods, styles of study, and/or degrees of certainty than would typically characterize a study of the same issue within the bounds of a single research discipline (Parson, 1995; Laniak et al., 2013). In order to make the complexity surrounding integrated assessment studies more tractable, a need has also arisen to integrate computer models from diverse fields so that scientists can conduct more holistic assessments. In particular, for water resources specialists, the need for model integration arises frequently because, although many individual hydrologic processes have existing mathematical models that are able to simulate them under a given set of circumstances, there is rarely a single model that can simulate all of them at the different scales and complexities desired while accounting for feedbacks between the various sub-processes for integrated assessment studies (Beven et al., 1980; Argent et al., 1999). Selecting a particular hydrologic model requires a consideration of the specific management challenges of concern, the spatial and temporal scales of interest, model input data availability, and computing requirements, among other considerations (Beven et al., 1980; Leavesley et al., 2002; Argent, 2004; Voinov and Shugart, 2013; Chowdhury and Eslamian, 2015; Clark et al., 2015). When a single model cannot meet the needs of a modeling study, it is common for modelers to couple elements of multiple models together to form a more holistic or accurate representation of a water system. This coupling must not only be considered as a technical exercise of stitching models together but must also include semantic and conceptual harmonization between coupled models (Janssen et al., 2011; Voinov and Shugart, 2013).

Although model developers in the earth systems and environmental modeling field have used several approaches to couple models for their integrated modeling efforts, the component-based modeling approach is increasingly receiving more attention. This is because, in contrast to monolithic approaches where models are compiled into a single code base or executable unit, component-based model development promises improved flexibility in the selection of the best available modeling approach for each application domain and re-use of different models, and more maintainable and extensible models (Fröhlich and Franz, 1999; Szyperski, 2002). The component-based modeling paradigm involves the provision of interface definitions describing standard data structures and functions that models must implement so that they can be deployed independently to exchange information at runtime with other models. Model developers across diverse fields who adopt these interface definitions can develop models that can be coupled with other models to simulate complex earth and environmental systems.

* Corresponding author. 8200 Old Main Hill, Logan, UT, 84322-8200, USA.
E-mail address: caleb.buahin@usu.edu (C.A. Buahin).

https://doi.org/10.1016/j.envsoft.2018.07.015
Received 13 December 2017; Received in revised form 20 April 2018; Accepted 25 July 2018
Available online 01 August 2018
1364-8152/ © 2018 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license
(http://creativecommons.org/licenses/BY/4.0/).

Software Availability

Name of software: The software described in this paper includes the HydroCouple component-based modeling interface definitions, the HydroCouple software development kit (SDK), and the HydroCoupleComposer component coupling graphical user interface (GUI) and command line interface (CLI). The two HydroCouple compliant components we developed for this study include a two-dimensional hydraulic model component called the Finite Volume Hydrologic Model (FVHM) and a component developed from the Environmental Protection Agency's Stormwater Management Model (SWMM) code.
Developer: Caleb A. Buahin
Contact: Caleb A. Buahin; Address: 8200 Old Main Hill, Logan, UT 84322-8200, USA; Email: caleb.buahin@usu.edu
Required hardware and software Any computer that runs the Windows, Linux, or Macintosh operating system. The performance of the SWMM and FVHM components benefit from compilation with a compiler that supports OpenMP (http://www.openmp.org). FVHM can be compiled with the Message Passing Interface (MPI) enabled. Compilation of all the software codes enumerated requires the Qt 5.7 framework (http://www.qt.io).
Cost The HydroCouple interface definitions, software stack, and model components in this manuscript are freely available under the GNU Lesser General Public License (LGPL) and can be downloaded from the HydroCouple Github repository (https://github.com/hydrocouple).

Component-based modeling provides a natural avenue for experimenting with different model formulations since model components can be removed and added to a composition in a "*plug-and-play*" fashion.

Several component-based modeling frameworks and interface standards with varying degrees of complexity and application domains have been developed over the years. These include the Earth Systems Modeling Framework (ESMF, Hill et al., 2004), Community Surface Dynamics Modeling System (CSDMS, Peckham et al., 2013), the Object Modeling System (OMS, David et al., 2002), and others. In the water resources modeling arena, the Open Modeling Interface (OpenMI, Moore and Tindall, 2005) definitions have been tested and used extensively (e.g., Smolders et al., 2008; Castronova and Goodall, 2009; Goodall et al., 2011; Buahin and Horsburgh, 2015). The appeal of OpenMI revolves around the fact that instead of providing a framework that has an accompanying large and complicated software stack that is beyond the expertise of many water resources modelers, the OpenMI developers provide a set of standardized, programming language agnostic interface definitions that can be adopted to develop model components that can communicate with each other directly at runtime. The OpenMI interface definitions are object oriented, with clear and well-defined inheritance relationships. Another major advantage is that the latest OpenMI 2.0 version has been adopted as an Open Geospatial Consortium (OGC) standard (Vanecek and Moore, 2014), which means that it has been reviewed and vetted by a large community of modelers.

Despite these attractive features, our experience using OpenMI revealed a few areas where advancements to the interface definitions would be useful, especially for water resources modeling applications. First, while the OpenMI interface definitions are programming language agnostic, the example interface definitions as well as software development kits (SDK) provided by the OpenMI developers use the C# and Java programming languages. These languages are compiled into an intermediate bytecode before being translated into the native instructions of a target machine to be executed using a virtual machine software infrastructure (i.e., Common Language Runtime (CLR) for C# and Java Virtual Machine (JVM) for Java). Additionally, C# has traditionally been restricted to computers that run Windows operating systems.

Conversely, many legacy model codes used in the water resources modeling field, and in the earth systems and environmental modeling field more generally, have been developed using programming languages like Fortran, C, and C++ because of the mature set of tools, scientific libraries, and support for HPC simulations that are available with these languages. Additionally, these programming languages are employed for computational models because they are compiled directly into native instructions for a target machine (i.e., natively compiled) and, therefore, generally have lower memory footprints, faster performance, and can be compiled on many operating systems. While the advent of Just-In-Time (JIT) compilers has significantly improved the performance of languages like C# and Java, natively compiled languages remain faster for many applications and remain the benchmark for evaluating performance (Taboada et al., 2013). To convert legacy codes written using natively compiled languages into components that can be coupled loosely to other models, one needs to resolve the programming language mismatch between the interface definitions and the computational codes of these legacy models. Though there are ways to bridge this programming language mismatch (e.g., using Platform Invocation Service for C# and Java Native Interface for Java), the costs of marshalling data across this language divide can lead to increased memory usage and increased time for individual simulations. We encountered and quantified these costs in a previous study where we converted the Environmental Protection Agency's Stormwater Management Model (SWMM), which is written using the C programming language into an OpenMI compliant component using the C# OpenMI interface definitions for a spatial domain decomposition urban stormwater coupling exercise (Buahin and Horsburgh, 2015).

Second, while the OpenMI specification provides interface definitions for representing geometric primitives (e.g., points, lines, and polygons) and their associated time varying data, these definitions lack some of the more common geospatial dataset formats used by water resources modelers (e.g., meshes, vector datasets, rasters, etc.). Also, the geospatial interface definitions provided by the OpenMI specification lack the topological relationship information that is important for many water resources modeling applications. For instance, a hydrologist simulating flows in a river network will need to know which upstream tributaries flow into any selected river reach. These types of topological relationships are not explicitly supported by the OpenMI 2.0 standard but can be implemented by extending the OpenMI standard.

Finally, like other earth systems and environmental modelers, hydrologic modelers often embark on experimental simulations where the same model is executed multiple times with varied inputs (e.g., optimization, uncertainty assessment, calibration, etc.). These types of simulations fall into the so called "embarrassingly parallel" class of simulations and benefit from using high-performance computing (HPC) resources. Many research institutions provide access to computing clusters comprised of heterogeneous hardware configurations of multicore Central Processing Units (CPU) as well as graphical processing units (GPU) and Many Integrated Cores (MIC) architecture accelerators that can be used for more efficient computations. However, the current OpenMI standard provides little direction on how to take advantage of these increasingly ubiquitous HPC infrastructures for more efficient simulations.

The contribution of this paper is the presentation of a set of component-based modeling interface definitions called HydroCouple and its associated SDK and coupled model composition tools. HydroCouple uses the OpenMI 2.0 interface definitions as its foundation but advances new interface definitions to address the challenges enumerated in the preceding paragraphs and others. We chose to build from OpenMI because it has been tested and used within the water resources modeling domain and because it has recently been advanced as an OGC standard.

This manuscript represents a maturation of the core concepts and interfaces presented in an earlier conference proceedings paper (Buahin and Horsburgh, 2016), where we first described new spatiotemporal data structures and interfaces for supporting parallelized experimental simulations like calibration, optimization, and uncertainty assessment within component-based modeling frameworks. In this paper, we extend the previous work by providing a more complete description of the spatiotemporal data structures and topological relationships supported by HydroCouple, by describing implementation of HydroCouple's interface definitions and software development kit using C++, by introducing HydroCouple's GUI for composing coupled models, and by providing a detailed case study that demonstrates how HydroCouple's new capabilities can be used in complex model coupling scenarios.

We also illustrate how these advancements can be used to convert existing, legacy codes into model components and develop new ones from scratch through a detailed modeling case study involving parallel execution of coupled models having different dimensionality. We coupled a one-dimensional (1D) hydraulic model developed from the Environmental Protection Agency (EPA) Stormwater Management Model (SWMM) and a two-dimensional (2D) hydraulic model that we developed called the Finite Volume Hydrologic Model (FVHM). This coupled urban stormwater modeling example illustrates how these advancements can help usher more water resources modelers into the HPC realm so that they can embark on more efficient simulations.

The remaining chapters of this manuscript are organized as follows. In Section 2, we provide a review of existing component-based frameworks with respect to their support for spatiotemporal data structures, data exchange workflows, and support for simulations on HPC infrastructure. In Section 3, we provide a discussion on the design of the HydroCouple framework with respect to the three areas identified in Section 2. We also describe tools provided to simplify the process of developing new components and creating coupled model compositions. In Section 4, we present the case study involving the coupling of 1D hydraulic model component and a 2D component developed to represent an urban stormwater conveyance system. Through this application we illustrate how the advancements we have implemented are used.

2. Design of component-based modeling frameworks

The component-based modeling approach and its precursor, the component-based software development approach, have their origins in the object-oriented programming approach with its notions of re-use through encapsulation, inheritance, and polymorphism. Components are typically paired with software frameworks, and they serve to extend the capabilities of frameworks, while frameworks provide an environment for executing components (Fröhlich and Franz, 1999). This concept extends to model components and modeling frameworks in earth systems and environmental modeling. For instance, CSDMS components are executed within the Common Component Architecture Fast Framework Example In Need of Everything (CCAFFEINE, Allan et al., 2002). The definition of standard interfaces forms the basis of interaction between components and frameworks and between components themselves by describing the assumptions they make about each other (Fröhlich and Franz, 1999). The design of OpenMI deviates from other approaches for developing component-based models by forgoing the pairing of interface definitions for components with a software framework. Instead, it prescribes interfaces that let components communicate directly with each other independent of a framework. This design choice was made to give more flexibility to component developers to optimize the data exchange process between model components.

The interface definitions for components are specified in a number of ways for different frameworks depending on the programming languages supported by the framework. Programming languages that were developed primarily for object-oriented programming like C++, C#, and Java have formal ways of specifying interfaces so that they can be inherited and implemented with details of their functioning to create framework compliant components. For example, C++ interfaces are specified in header files as classes with only pure virtual functions. The OpenMI interfaces for C# and Java are specified this way. On the other hand, for component-based modeling frameworks that support languages like C and Fortran (e.g., ESMF and CSDMS), models are required to register pointers to their standard functions with the framework. These standard function pointers are then stored in virtual function tables that can be accessed by other components to achieve the interfacing functionality. While the OMS framework was written using Java, which is an object-oriented programming language, the OMS interfaces are specified by marking classes, functions, and fields of a component with standardized java annotations (e.g., @In, @Out, @Execute, etc.), which serve as a form of syntactic metadata. Through Java's reflection capabilities, annotated classes, fields, and functions can be accessed and invoked at runtime.

The typical core interfaces defined for models in component-based modeling frameworks are the *initialize*, *run*, and *finalize* (IRF) functions (Syvitski et al., 2011). The *initialize* interface function is implemented to instantiate the resources and inputs needed by a model for a simulation. The *run* interface is responsible for performing the underlying computations of a model (e.g., performing a time-step). The *finalize* interface is implemented to dispose of the computational resources used by the model (e.g., closing output file streams, deallocating memory, etc.). In addition to these core interfaces, models also specify interfaces for the types of inputs and outputs that can be consumed and shared with other components respectively. CSDMS attempts to standardize these elements that are shared by various component-based modeling frameworks by providing a core set of interface definitions called the Basic Modeling Interface (BMI) definitions (Peckham et al., 2013).

Table 1 summarizes the design considerations we identified for component-based modeling frameworks and standards used in the earth

Table 1
Design considerations for component-based modeling frameworks.

Design Consideration	Description
(1) Cross-platform support	Can be compiled and executed on multiple operating systems
(2) Limit Framework Dependence	Direct data exchange between components with minimal intervention from the underlying framework
(3) Language interoperability	Developed using an efficient programming language that provides bindings with several of the programming languages widely used in developing models
(4) Support for simulations on HPC infrastructure	A framework that supports simulations that take advantage of increasingly ubiquitous of distributed and heterogenous computing infrastructure with simple and transparent interfaces
(5) Support for standard spatiotemporal datasets	Supports widely used open standards for representing geospatial datasets
(6) Flexible and customizable data exchange workflows	Data exchange workflows that are flexible and customizable while minimizing the amount of coding needed to achieve this flexibility and preventing code re-compilation
(7) Development, model composition, and visualization tools	Provides comprehensive model development tools, coupled model compositions tools, and visualization tools to minimize the expense of model development

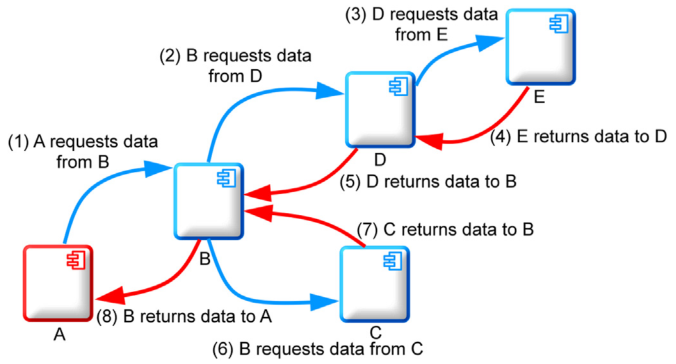
(3) D requests data from E
(2) B requests data from D
(1) A requests data from B
E
(4) E returns data to D
D
(5) D returns data to B
(7) C returns data to B
B
A
(8) B returns data to A
C
(6) B requests data from C

Fig. 1. OpenMI "request and reply" data exchange mechanism. Component A is the controller/trigger for the simulation.

systems and environmental modeling arena. In the sections that follow, we discuss three of these design considerations that we believe are important areas where we identified opportunities for improvements to be made. We focus on their support for spatiotemporal datasets, options for data exchange workflows between components at runtime, and support for simulations on HPC infrastructure. Given that OpenMI was the base from which we built HydroCouple, we discuss how OpenMI is designed with respect to these three areas and contrast it with the design of other component-based modeling frameworks. Limitations in these three areas are impediments to using OpenMI and, more generally, component-based modeling in practice because important geospatial data structures used by many models are not currently supported, data exchange workflows are restrictive or require re-compilation for each coupled model composition, and HPC simulations are either complex or not supported at all by existing frameworks.

2.1. *Definition of spatiotemporal data structures*

The types of inputs that can be consumed and the outputs that can be supplied by models in a component-based modeling framework are typically organized as multi-dimensional arrays of data that can be accessed using indexes along their respective dimensions (e.g., ESMF, CSDMS, OpenMI). These inputs and outputs are often further abstracted into domain specific types for many component-based modeling frameworks. For instance, OpenMI provides a time-space input/output specialization that associates geometric primitives including points, polylines, polygons, and polyhedra with time varying data. For hydrologists, this could be used to a represent hydrologic feature like a river network and its associated time varying flows.

In water resources modeling applications, these features are often used for delineating a model's spatial domain and prescribing boundary or input data for models - e.g., polygons for watershed boundaries, polylines for alignments of rivers, river cross-sections, etc. Missing from the OpenMI definitions is the topological information that provides the spatial relationships between adjacent geometries. Yet, this information is critical for many applications. For example, although the individual cells of a two-dimensional computational grid used for a hydrodynamic model of a reservoir may be represented by a list of polygons, the adjacency information between cells that is needed to numerically approximate spatial gradients of variables are missing. On the other hand, the CSDMS suite of tools for component-based modeling, which were developed for ice, terrestrial, coastal, and marine applications as well as ESMF, which was developed for global weather and climate predictions, focus on providing interfaces for gridded datasets, including logically rectangular, unstructured, and curvilinear grids in one, two, and three dimensions. Explicit support for geospatial data types like those provided in OpenMI are, however, not supported.

2.2. *Data exchange workflows*

The model execution and data exchange workflow between components in a component-based model composition is handled in a variety of ways for different frameworks. ESMF requires a modeler to write a driver program called an "*AppDriver*" that contains the "main" routine for an ESMF application (Collins et al., 2005). This "*AppDriver*" needs to be compiled for each application and is responsible for directing the time-stepping and the data exchange between components. The drawback with this approach is that a new driver needs to be written and compiled for each composition. Therefore, a modeler will have to be a programmer with the know-how and compilation tools to compile the application, putting it out of reach for many water resources modelers who are typically not expert programmers.

In contrast to ESMF, OMS provides a way for modelers to direct model execution and data exchange between components externally. This is accomplished by letting users write the business rules for a simulation using a OMS prescribed domain specific language (DSL) specified in an external file (David et al., 2013). In contrast to a general-purpose programming language, a DSL is a relatively simple specification language dedicated to a particular problem domain, a particular problem representation approach, and/or a particular solution technique (Deursen and Financial, 1997; Deursen et al., 2000; David et al., 2013). The benefit of this setup is that model users are able to direct the data exchange between components without recompiling the entire code or being restricted to the in-built workflows provided by a component-based modeling framework.

The primary model execution and data exchange mechanism in OpenMI is based on the pull-based pipe and filter architecture (Buschmann, 1996). With this method, components exchange memory-based data directly without an external data exchange workflow controller using a "request and reply" mechanism (Gregersen et al., 2007). The most downstream component in a composition is designated as the controller/trigger for an entire simulation where it requests the data it needs from upstream components that it is coupled to and waits for the data it requests to be returned before proceeding with its computations. Upstream components issue their own requests to their respective upstream components in a cascading fashion and wait to receive the data they requested before performing their computations as illustrated in Fig. 1. Data exchange between an input (defined by the *IBaseInput* interface) and an output (defined by the *IBaseOutput* interface) can be mediated by an adapter (defined by the *IBaseAdaptedOutput* interface) that performs the necessary data transformations (e.g., temporal interpolation, spatial interpolation, unit conversion, etc.) that are needed to supply the correct requested data from one component to another. Contextual adaptors are generated by a factory interface definition called an *IBaseAdaptedOutputFactory* that uses the input and output that are to be mediated as query variables to generate appropriate adaptors.

While the "request and reply" data exchange approach requires no external data exchange controller and works for many applications, it does not work for those compositions that have two or more downstream components as illustrated in Fig. 2. In Fig. 2, components D and E are not executed since they are not involved in the request and reply chain of the trigger component, A. Yet, there are circumstances where they may be needed as part of a coupled model composition. For example, component A might be a stream temperature model, B might be a streamflow model, and E might be a contaminant transport model. Both E and A need streamflow data from B, but E would never get executed if A is the triggering component.

The OpenMI developers recognized that the pull-driven data exchange approach might be too restrictive for certain applications. Therefore, they suggested a loop driven data exchange workflow where the coupled system will loop over all components to let them check if they need to take action before proceeding (Moore, 2010). The loop approach has been included in the interface standard of OpenMI 2.0, but the implementation for this loop driven approach is not provided as

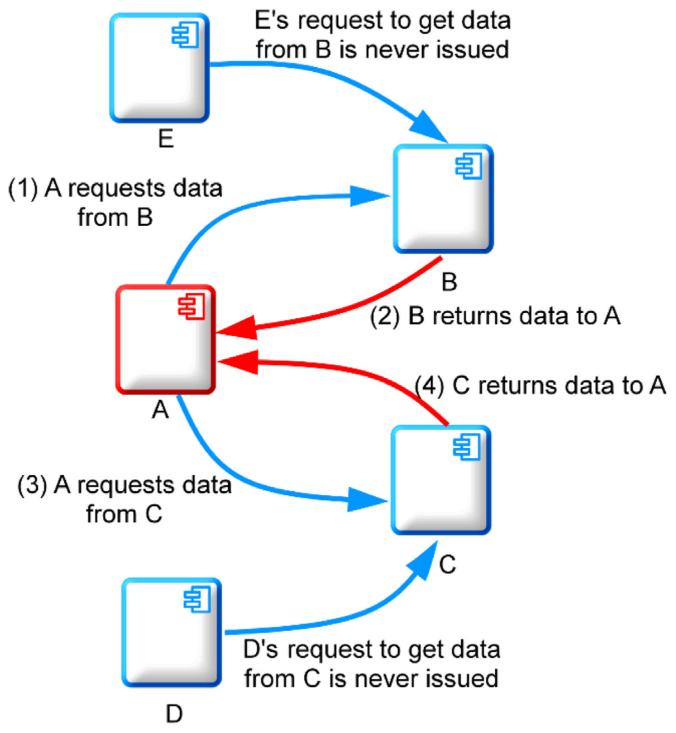
E's request to get data from B is never issued
E
(1) A requests data from B
B
(2) B returns data to A
A
(4) C returns data to A
(3) A requests data from C
C
D's request to get data from C is never issued
D

Fig. 2. Limitation of the OpenMI "request and reply" data exchange mechanism for compositions with more than one downstream component.

part of the OpenMI SDK. In contrast to ESMF and OMS, OpenMI may be easier to use for non-expert programmers. However, the fact that the request and reply mechanism used by OpenMI may not work for all model compositions and the fact that the loop approach has not been implemented in OpenMI's SDK means that OpenMI could be further improved to meet additional use cases.

2.3. Support for simulations on HPC infrastructure

Increasingly, the complexity of the hydrologic modeling applications that modelers embark on requires that model execution be conducted using distributed computing techniques on HPC infrastructure. HPC infrastructure typically involves pooling of computational resources from many computers with multi-core processors that are networked together. ESMF was designed to support both data and task parallelism on HPC infrastructure by making ESMF components the very units of parallel execution to aid model developers in writing highly efficient and scalable codes (Collins et al., 2005). A virtual machine (VM) approach is adopted to abstract away the details underlying model execution in HPC environments using the Message Passing Interface (MPI) standard. Each component in an ESMF composition is assigned a single VM comprised of one or more persistent execution threads (PET), where each PET is equivalent to a single MPI process. A component may allocate its PETs to child components. Additionally, a developer can employ fine grained, shared memory parallelism within a PET using OpenMP or other multi-threading approaches. Inter- and intra- VM communications are handled by the framework in a fashion similar to MPI where data must be provided as raw, language specific, one-dimensional, contiguous arrays.

OMS adopts a similar approach to ESMF to achieve parallelism by making components themselves the units of parallel execution. However, unlike ESMF, OMS only supports shared memory parallelism applications using multi-threading, where each model component is executed in its own separate thread. In contrast, OpenMI provides little direction on how to take advantage of HPC resources for more efficient simulations. For scenarios where a downstream component requests input data from one or more upstream components, an OpenMI component may make these requests using parallel threads in a multi-threading environment for efficiency. However, this approach only scales up to the number of upstream components that supply data to a single component. Additionally, in a bidirectional data exchange scenario between two coupled model components, A and B, OpenMI's request-reply data exchange mechanism forces a sequential execution of the components involved. Component A requests the data it needs from B. However, component B needs data from component A before it can execute. To avoid an infinite recursion loop, component A provides an estimate of the data requested by component B so that component B can advance the computation. From a performance perspective, it may be desirable to execute both model components concurrently using estimates of data provided from the component that is coupled to them.

To the best of our knowledge, existing component-based modeling frameworks lack support for automated parallel simulations for those experimental evaluations that are "embarrassingly parallel" involving executing the same coupled model instances repeatedly with varied input parameters - e.g., automated parameter estimation and uncertainty assessment (e.g., Blasone et al., 2008), optimization (e.g., Zhang, 2009), ensemble simulations (e.g., Komma et al., 2007), etc. While, these types of applications can be currently undertaken within existing component modeling frameworks, they can either only be configured to run sequentially, require manual intervention, or require writing code outside of the scope of a component-based modeling framework.

3. Design of the HydroCouple framework

To address the challenges discussed in the preceding sections, we developed the HydroCouple component-based modeling interface specifications and associated software. HydroCouple builds on the strengths of OpenMI while advancing new interface definitions to deliver standard spatiotemporal dataset interfaces with their associated topological information and providing support for more efficient simulations on HPC infrastructure. Like OpenMI, the HydroCouple interface definitions are language agnostic. However, we implemented the HydroCouple interface definitions using C++ because in addition to the benefits enumerated in the previous sections (i.e., cross-platform support, relatively lower memory footprints, mature set of tools and scientific libraries, etc.), it also provides bindings with many programming languages and serves as a good foundation for a framework that seeks to integrate legacy model codes and support models written using different programming languages. The interfaces for HydroCouple have been provided in four C++ header files with about 1500 lines of code that provide the core interface definitions, a spatial extension, a temporal extension, and a spatio-temporal extension. In the following section, we discuss the key choices that were made in the design of HydroCouple and the motivation behind them.

3.1. Types of HydroCouple components

The core interface definition of the OpenMI standard is the *IBaseLinkableComponent* interface. This interface encapsulates the computational engine of a model and defines the IRF interfaces. It is also responsible for defining the types of inputs a component can consume and the type of outputs a component can supply to other components. In HydroCouple, the *IBaseLinkableComponent* interface has been superseded by the *IComponentInfo* interface definition as the core interface definition. This interface definition allows model developers to provide details about a component including: documentation about the formulations employed in the component, limitations of the component, coupling configurations that can be employed, data transformation adaptors that can be employed, as well as details about the coupling configurations and application scenarios for which the use of the component is appropriate. Other details that can be specified through this interface include: the name of the developer of the component, contact information for the developer, classification of the component to properly categorize component libraries in a GUI, and component version. These details, which can be used by modelers to seek further assistance are currently missing for components developed for many frameworks leading to poor understanding about their capabilities and their proper usage in component-based modeling applications. Preferably, these details should be specified in an external file and read by the component at runtime for display purposes.

In contrast to OpenMI, which only allows a single component type, instances of classes based on the *IComponentInfo* interface create three types of specialized components. These specializations include classes that are based on the *IModelComponentInfo*, the *IWorkflowComponentInfo*, and the *IAdaptedOutputFactoryComponentInfo* interfaces as shown in Fig. 3. Instances of classes based on the *IModelComponentInfo* are responsible for creating instances of classes based on the *IModelComponent* (Fig. 4) interface, which is equivalent to the OpenMI specification's *IBaseLinkableComponent*. Instances of classes based on the *IAdaptedOutputFactoryComponentInfo* interface are responsible for creating instances of a new component type based on the *IAdaptedOutputFactoryComponent* interface (Fig. 4). This new interface definition can be implemented as an independently deployable component that is equivalent to OpenMI's *IAdaptedOutputFactory* interface definition. However, unlike the OpenMI definition, it is not bound to any particular component and can be reused between different models to generate adaptors for data exchange mediation. Finally, instances of classes based on the *IWorkflowComponentInfo* interface are responsible for generating another new type of component based on the *IWorkflowComponent* interface (Fig. 4) that is responsible for managing simulations and the data exchange workflow between components. These new component types have been added to HydroCouple to avoid the duplicative work of developing new data exchange workflows and data adaptors for each component or coupled modeling application when using OpenMI and component-based modeling frameworks in general. For example, a workflow component could be developed to automatically discover the dependencies that exist between model components and execute them in a sequence that ensures that all components are able to obtain the data they require. A time series interpolation adapter could be developed as an independently deployed component that can be reused by different model component compositions. This contrasts with the current OpenMI setup where an adaptor is bound to a component and needs to be implemented for every model component that needs to use it. This generalizes and formalizes this process so that adaptors can be developed into components in their own right.

3.2. HydroCouple spatiotemporal data structures interface definitions

In order to reduce the cost of converting legacy models into components, it is important that the low-level geospatial data structures that hydrologists widely use are made available in component-based modeling frameworks. To achieve this goal, HydroCouple provides an explicit implementation of the Open Geospatial Consortium's (OGC) Simple Feature Access (SFA) specification (Herring, 2011). In addition to providing interfaces to describe various types of geometries, the advantage of implementing the OGC SFA is that it defines functions for performing topological queries (e.g., checking for intersection between features, checking if one feature touches another, etc.) and geospatial operations (e.g., unions, buffering, symmetric difference, etc.) that are useful in water resources modeling applications. For instance, an intersection operation can be used between gridded precipitation output from a weather forecast model and the boundary of a watershed in a hydrologic model to estimate the fraction of each grid cell that contributes precipitation flux to that watershed.

HydroCouple additionally prescribes interface definitions for various gridded dataset types for hydrodynamic and hydraulic modeling applications as well as representing some other gridded dataset types often utilized in the water resources area. For example, HydroCouple explicitly defines an interface for multi-banded raster datasets that includes the number of raster grid cells, cell size, data type, no data value, etc. This interface can be employed to represent various types of datasets in models, including digital terrain models (DTM), aerial imagery, time varying land use data, etc. In addition to this definition, HydroCouple defines interfaces for cartesian, rectilinear, and curvilinear grids in two and three dimensions for hydraulic and hydrodynamics modeling applications. Time-varying data for these grids may be associated with the nodes, faces/edges, and volume/area of their individual cells. For networks, unstructured meshes, and polyhedra, HydroCouple adopts the quad edge data structure proposed by (Guibas and Stolfi, 1985). With this data structure, a directed edge stores the complete topological information about the resulting polyhedra/network by storing pointers to its left and right face polygons and its origin and destination vertices.

These spatiotemporal interface definitions have been provided to abstract away the details of the various types of the same data structures provided by various software libraries. For example, a raster dataset can be stored as a GeoTIFF file or a NetCDF file which are accessed using different software libraries. This was done so that data can be accessed within HydroCouple using common interfaces, regardless of the underlying file format of the data. A companion SDK was also developed to enable reading and writing of spatiotemporal data to and

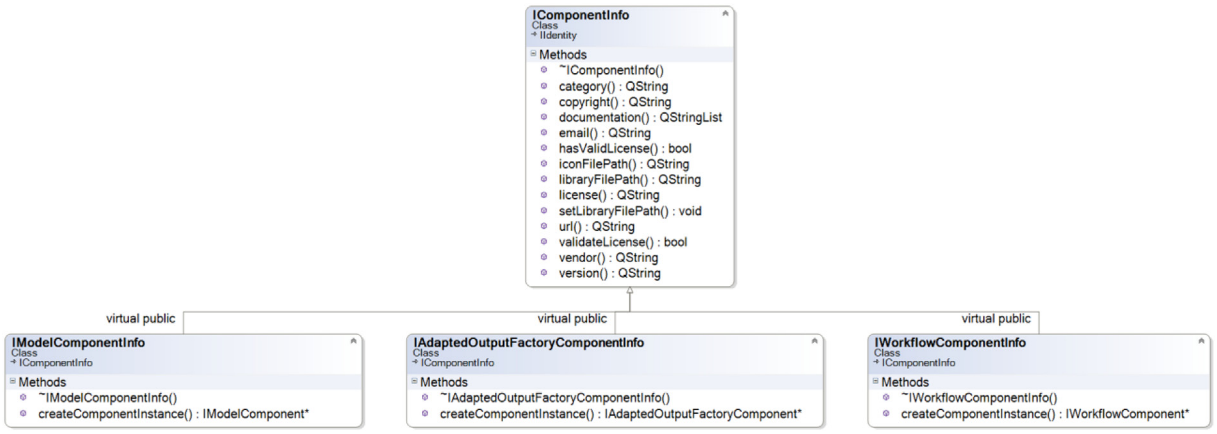

Fig. 3. UML diagram for the IComponentInfo interface.

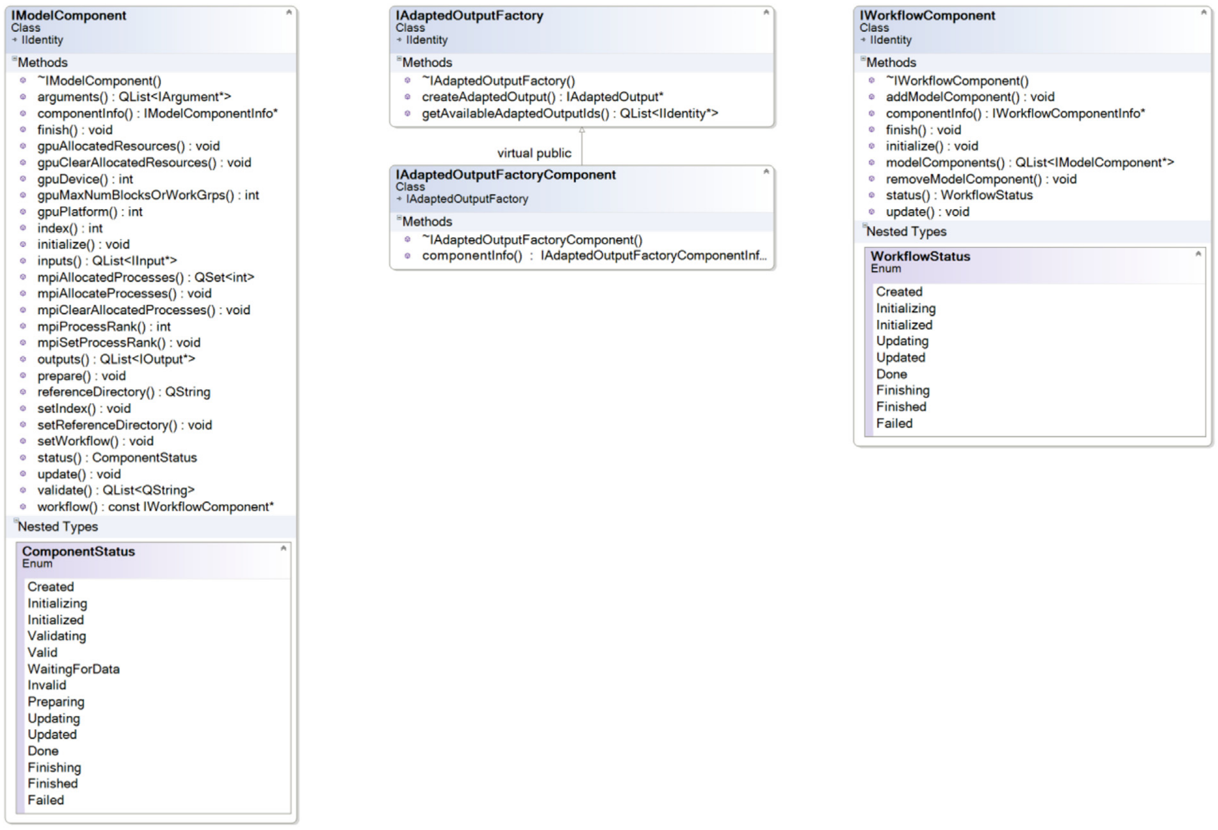
IModelComponent
Class
+ IIdentity
Methods
~IModelComponent()
arguments() : QList<IArgument*>
componentInfo() : IModelComponentInfo*
finish() : void
gpuAllocatedResources() : void
gpuClearAllocatedResources() : void
gpuDevice() : int
gpuMaxNumBlocksOrWorkGrps() : int
gpuPlatform() : int
index() : int
initialize() : void
inputs() : QList<IInput*>
mpiAllocatedProcesses() : QSet<int>
mpiAllocateProcesses() : void
mpiClearAllocatedProcesses() : void
mpiProcessRank() : int
mpiSetProcessRank() : void
outputs() : QList<IOutput*>
prepare() : void
referenceDirectory() : QString
setIndex() : void
setReferenceDirectory() : void
setWorkflow() : void
status() : ComponentStatus
update() : void
validate() : QList<QString>
workflow() : const IWorkflowComponent*
Nested Types
ComponentStatus
Enum
Created
Initializing
Initialized
Validating
Valid
WaitingForData
Invalid
Preparing
Updating
Updated
Done
Finishing
Finished
Failed
IAdaptedOutputFactory
Class
+ IIdentity
Methods
~IAdaptedOutputFactory()
createAdaptedOutput() : IAdaptedOutput*
getAvailableAdaptedOutputIds() : QList<IIdentity*>
virtual public
IAdaptedOutputFactoryComponent
Class
+ IAdaptedOutputFactory
Methods
~IAdaptedOutputFactoryComponent()
componentInfo() : IAdaptedOutputFactoryComponentInf...
IWorkflowComponent
Class
+ IIdentity
Methods
~IWorkflowComponent()
addModelComponent() : void
componentInfo() : IWorkflowComponentInfo*
finish() : void
initialize() : void
modelComponents() : QList<IModelComponent*>
removeModelComponent() : void
status() : WorkflowStatus
update() : void
Nested Types
WorkflowStatus
Enum
Created
Initializing
Initialized
Updating
Updated
Done
Finishing
Finished
Failed

Fig. 4. UML diagram for the HydroCouple's new component types.

from files on disk using multiple file formats. Details of the implementation of these interfaces within the SDK using existing libraries are provided in subsequent sections.

3.3. HydroCouple data exchange workflow interface definitions

Following the OMS approach where the data-exchange workflow can be specified externally, the *IWorkflowComponentInfo* interface definition was introduced in HydroCouple as a way to provide modelers with flexibility in prescribing the data exchange workflow in a component-based application. For example, OpenMI currently supports only the "request and reply" data exchange approach, whereas HydroCouple adds the ability to create independent workflow components that can execute using a loop-driven data-exchange approach as well as others. This interface creates an instance of an *IWorkflowComponent* that is responsible for keeping track of all components involved in a simulation and coordinating their computations and data exchange until the completion of a simulation. Just as the *IModelComponent* interface has the IRF functions, we defined the *IWorkflowComponent* with these same functions to enable initializing the workflow component, updating components associated with the workflow by asking them to perform their computations, and finalizing the components upon completion of a simulation. If the *IWorkflowComponent* interface is not instantiated for a simulation, all components must default to the original OpenMI "request and reply" approach to data exchange. This new interface was added so that modelers can directly control data exchange in a component-based modeling simulation beyond the "request and reply" data exchange mechanism provided by OpenMI. Like model components, a suite of independently developed and deployed workflow components may be developed so that model users are able to select and integrate the workflows that are most suitable for their simulations.

3.4. HydroCouple on HPC infrastructure

HydroCouple supports parallelized, experimental simulations by introducing a new interface definition called the *ICloneableModelComponent* interface that inherits from the *IModelComponent* interface. This new interface introduces a *clone* function that must be implemented for a component to make a copy of itself. The *clone* function has been added so that independent copies of a model instance can be made for parallelized simulations. Details of the cloning approach are left up to the model component developer. A parent model component keeps track of all of its child clones, which can be accessed using the *children* function. Linkages with other model components are left up to the caller of the *clone* function. This cloning process may involve making a copy of the parent *ICloneableModelComponent* class and initializing it with the same arguments as the parent while making sure that outputs from the parent and child do not conflict. An example optimization application of this cloning feature is described in Buahin and Horsburgh (2016).

HydroCouple was designed to support simulations on HPC infrastructure by supporting parallel simulations on both shared and distributed memory systems as well as providing support for simulations that use general purpose computing on graphics processing units (GPGPU). Support for distributed memory parallel computing was designed using the Message Passing Interface (MPI) standard, while the GPGPU parallelism was designed to support the Open Computing Language (OpenCL) and Nvidia's CUDA framework. The *clone* function described is only one of the ways to enable parallelism in HydroCouple. Like OpenMI, HydroCouple provides interface definitions that must be implemented by a model developer. Component developers must implement all the MPI, OpenMP, or GPGPU functions that enable parallel computations. The SDK we developed (Section 3.5) provides some resources that can be used to help in the implementation of these functionalities.

In designing HydroCouple's support for simulations on HPC platforms, we followed the original OpenMI design choice of limiting the role of the framework in mediating the data exchange between components. Along this line, HydroCouple applications adopt MPI for coarse grained parallelism, where all components involved in a coupled modeling application are initialized and coupled directly on the MPI task/process with rank 0 (i.e., the master MPI Process 1) as illustrated in Fig. 5. For components that support MPI, a user may partition the remaining MPI processes to them to ensure the computational load is balanced (Fig. 5). This is done by specifying the MPI ranks available to each component in a coupled model application configuration file. Frameworks like ESMF automatically map computational resources into virtual machines that can be allocated to components for model execution on HPC resources. While convenient in abstracting away the complexity of computational resources available, it elevates the role the framework plays in mediating the data exchange between components. Additionally, getting up to speed with the large software code stack and its accompanying complexity is beyond the expertise of many water resources model developers. HydroCouple uses relatively few lines of code and distributes HPC resources directly by dividing available MPI ranks among components. Each component can group its allocated MPI ranks into an MPI communicator that can be shared among the component and its children. This approach can be more transparent for users of shared HPC infrastructure that use job scheduling software like Slurm and reduces the role of the framework.

Developers of components need not necessarily conduct their computations on the master MPI process. Components initiated on the master MPI process could be developed as proxy-stubs that serve as pathways for communicating results computed on child components on worker MPI processors. For example, in Fig. 5, the role of Component A, which is initialized on the master MPI process to communicate directly with Components B and C, may only involve collating computed results from its worker components initialized on MPI processes 2, 3, and 4 and sharing with other components. Once initialized, data sharing between components on different MPI processes can be conducted using standard MPI calls. Components on the master MPI processes are responsible for issuing messages to components on the worker processes to dispose themselves upon completion of a simulation. Fine grained parallelism may be employed over the CPU cores allocated to each MPI process using shared memory parallelism application programming interfaces (APIs) like OpenMP.

Like MPI, GPGPU frameworks abstract away the complexity underlying computing resources from users and present a single, virtual interface for accessing the hardware. To ensure that GPGPU resources are distributed efficiently between components, HydroCouple lets users prescribe the GPGPU device as well as the maximum number of CUDA blocks or OpenCL work groups each component can use on each MPI process. This gives users flexibility to partition jobs to GPGPU devices in a way that is tailored to the particular hardware layout on an HPC system. This partitioning of computing resources can be accomplished using job scheduling software like the Slurm Workload Manager that is used on many high-performance computing centers' systems.

3.5. HydroCouple composer and software development kit

In order to facilitate the development of components for HydroCouple, we developed a SDK that implements many of the core interfaces needed to develop a component as a software class that has initialization arguments, exchangeable inputs, exchangeable outputs. The SDK also implements the functions required to initialize a component by reading and validating initialization arguments and functions that describe the variables exchanged by component inputs and outputs. This enables model developers to focus on the computational portions of their code, which are placed within the correct locations inside the HydroCouple SDK classes. The core interfaces that have been implemented as classes include those for describing and identifying components and their inputs and outputs, the spatiotemporal domains of these inputs and outputs, the variable types consumed and supplied by these input and outputs, and the units of these variables. In HPC environments, the SDK automatically groups processors allocated to a component into a single private MPI communicator context so that communication between the component and its workers do not conflict with other components.

The SDK uses the Geospatial Data Abstraction Library (GDAL) (Warmerdam, 2008) extensively to provide support for reading various geospatial vector data formats (e.g., shapefiles, GeoJSON, GML, CAD, KML, etc.) into objects defined by OGC's SFA that can then be accessed

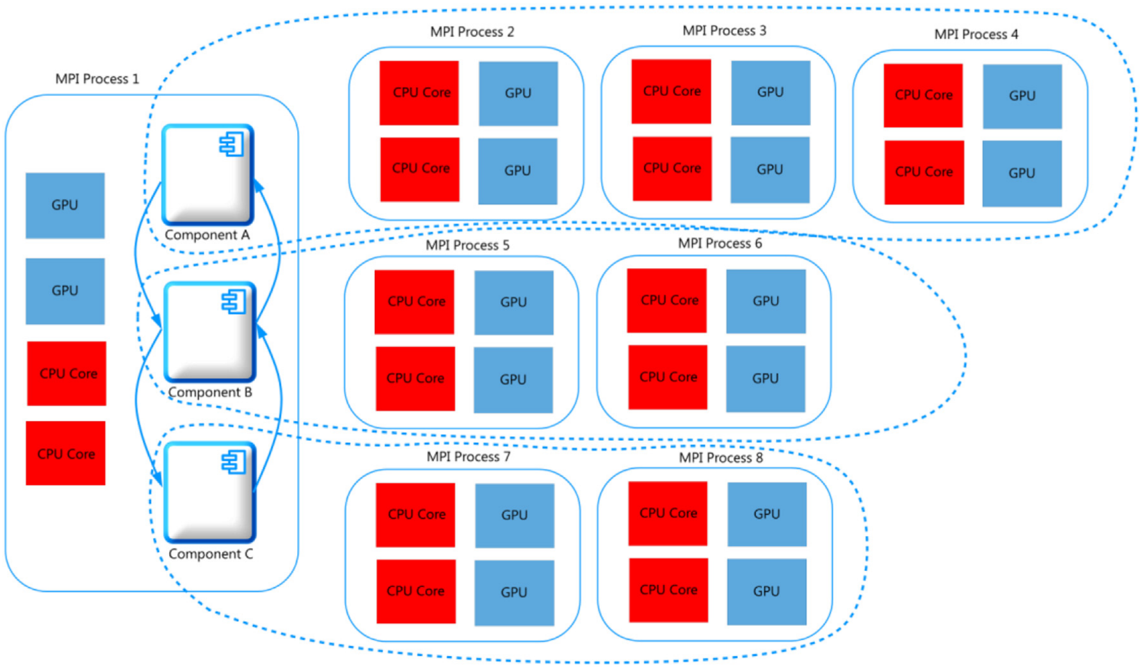

Fig. 5. Example HydroCouple configuration on high performance heterogeneous computing infrastructure where each component is assigned a number of GPU (blue) and CPU Core (red) resources.

programmatically by model components. The SDK also enables writing these objects to files on disk (e.g., for exporting model output). The HydroCouple SDK also uses the GDAL library to provide similar support for various raster dataset formats including the GeoTIFF, GRIB, SQLite, JPEG, and HDF5 formats. Work is also underway to implement the UGRID version 1.0 (http://ugrid-conventions.github.io/ugrid-conventions) convention, which is a proposal for storing unstructured mesh data in standard format using the Unidata Network Common Data Form (NetCDF) file using the Climate and Forecast (CF) metadata convention (Eaton et al., 2011) as a starting point.

In addition to the SDK, we developed the HydroCouple Composer graphical user interface (GUI) to provide model developers with a visual environment for composing coupled model configurations (Fig. 6). The HydroCouple Composer software displays all available model component libraries and allows users to drag and drop them onto to a graphical palette. Coupled model compositions are specified through a configuration file that lists the model components and their connection nodes as well as the workflow and adaptor components that are utilized. Each model component can be initialized using an initialization file that contains parameters specific to the component. Alternatively, components can be initialized using instances of a class based on the *IArgument* interface that are created in memory. After a component is initialized, compatible outputs and inputs from other components can be coupled interactively. Compatibility for coupling is

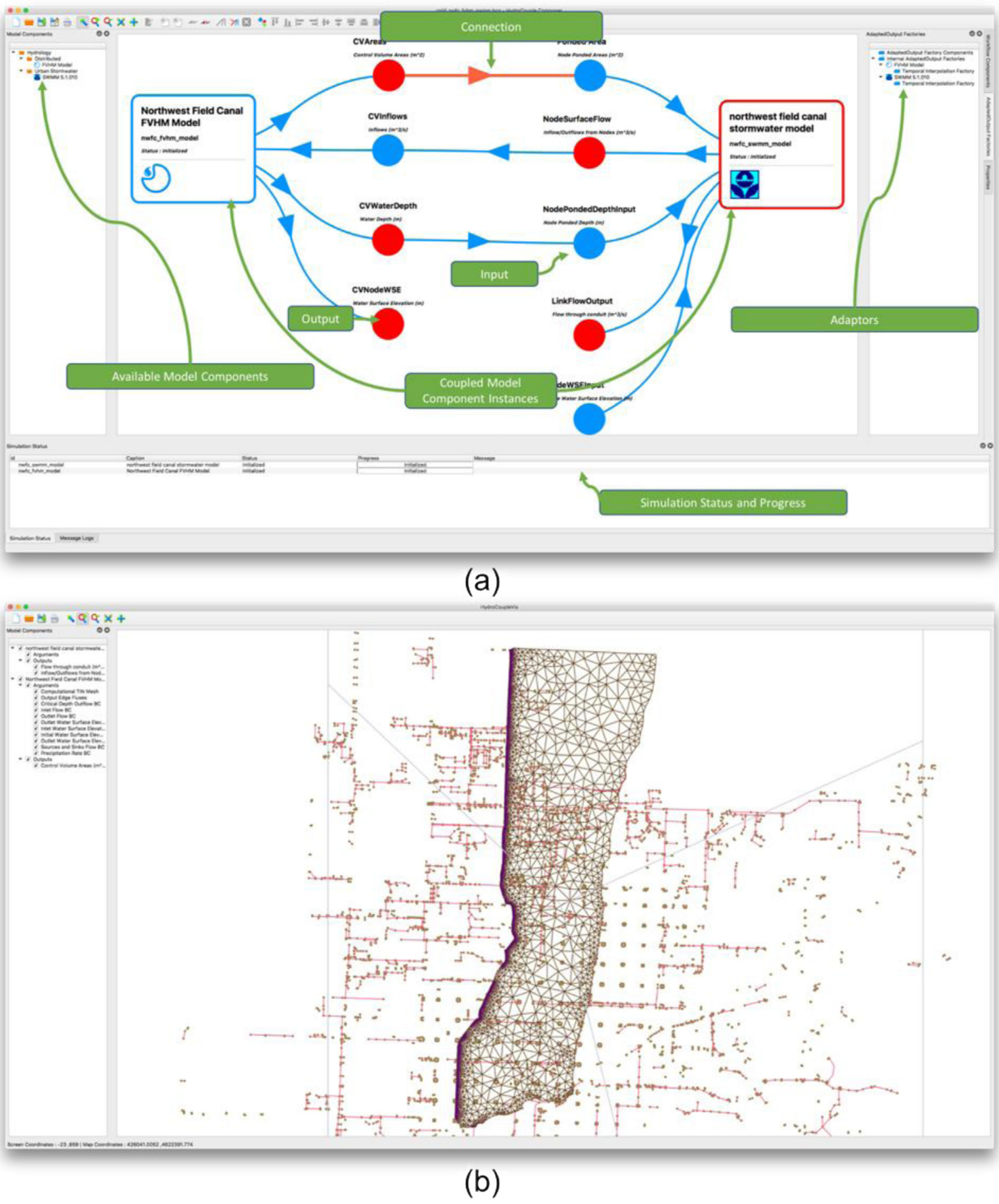

Fig. 6. HydroCouple Composer GUI: (a) coupled model composition canvas; (b) geospatial data visualization canvas.

determined by a function called *canConsume* on an instance of the input class that is to be coupled that is called when a model user tries to create a link between and input and an output. This function returns true or false indicating whether the coupling is valid based on business rules defined by the component developers. Selecting an existing connection between an input and output displays the available contextually relevant adaptors that can be inserted to mediate data exchange. For example, when a connection between time series input and output is selected, only time series related adaptors will be available for selection in the HydroCouple Composer GUI.

HydroCouple Composer is responsible for partitioning computing resources to model components based on a user's specifications in the configuration file and launches simulations. Example configuration files can be accessed in the HydroCouple GitHub repository (see Software Availability section). The HydroCouple Composer also monitors the progress of simulations and displays them to a user, logs messages from components, and provides rudimentary visualization capabilities (Fig. 6) for the standardized spatiotemporal inputs and outputs of model simulations. Finally, when the executable of the HydroCouple composer is launched using predefined command line arguments, it doubles as a command line interface for launching simulations on HPC resources. In Fig. 6, model components are shown as squares, connections between components are shown as directed arrows, inputs are shown as blue circles, and outputs are depicted as red circles.

4. Case study: coupling a 1D and a 2D hydraulic model using HydroCouple

In order to illustrate how HydroCouple's new interfaces facilitate the development of components and more efficient simulations, we applied them to couple a 1D hydraulic model component that simulates flow through pipes, culverts, inlets, outfalls, and other urban stormwater infrastructure with a 2D hydraulic model component that simulates flow in rivers, canals, and overland areas. The 1D hydraulic model component was developed using the EPA's SWMM model, while the 2D model component is a new formulation we developed called the Finite Volume Hydrologic Model (FVHM). These two models were specifically chosen to illustrate: 1) how the HydroCouple interface definitions and SDK can be used to convert legacy models into components; 2) how to develop new components from scratch; 3) how to handle the potential coupling configurations across models having differing dimensionality (i.e., 1D versus 2D); and 4) how coupled model components can be executed in parallel on HPC environments for more efficient simulations.

This particular type of 1D-2D hydraulic model coupling is widely implemented in the water resources modeling field because the trade-offs between 1D and 2D hydraulic models complement each other. 1D models are appropriate for simulating flows accurately and efficiently in channels, pipes, and other conduits with well-defined shapes. Although 1D hydraulic models are generally more computationally efficient, they are unable to accurately simulate lateral movement of flood waves into the floodplain, and they incorporate topography and bathymetry using cross-section profiles at various sections along the length of a river/pipe whose placements are relatively subjective (Samuels, 1990; Hunter et al., 2007). On the other hand, 2D hydraulic models are more suitable for simulating landscape processes and overland flows, albeit at a generally higher computational expense. Many urban hydrologic modeling scenarios (e.g., stormwater runoff, flooding, design of green infrastructure, and assessment of stormwater best management practices) require accurate representations of both the drainage network and the urban landscape, making a 1D-2D hydraulic model coupling ideal for simulating these scenarios. The 1D-2D coupled model discussed in this manuscript was developed for a subcatchment in the City of Logan, Utah. To illustrate the benefits of using HPC resources, we evaluated the simulation time speed up as more computational cores were allocated to the coupled model.

4.1. Study area

The City of Logan's stormwater conveyance system has its foundations in agricultural canals developed at the founding of the City to

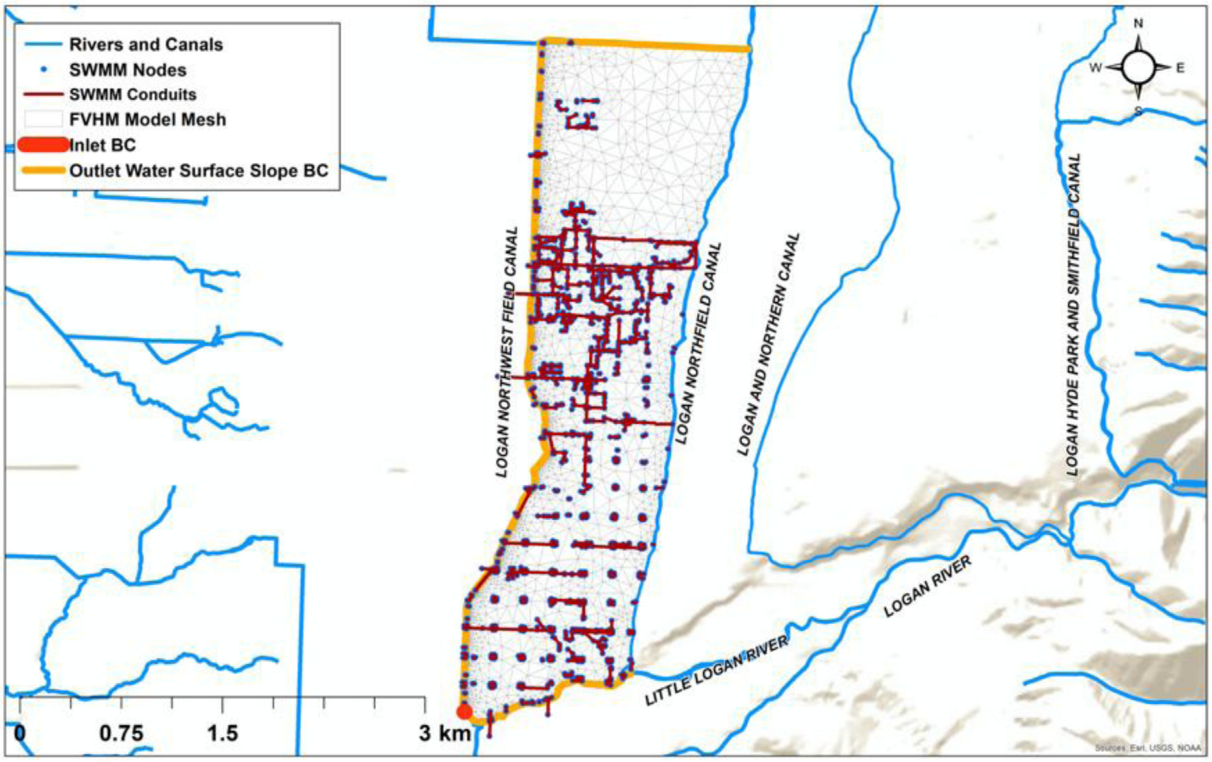

Fig. 7. 1D-2D coupled model boundaries.

divert irrigation water from the Logan River for farming. As the Logan River flows westwards through Cache Valley, it is diverted at various locations along its length into these unlined canals that flow northward. These same canals serve as the primary conduits for stormwater conveyance, with many of the City's stormwater outfalls piped directly into them. The boundary for the 5.81 km^2 area we used for our modeling study encompasses the area that drains into one such canal called the Northwest Field canal (Fig. 7).

4.2. SWMM model development

The SWMM code solves the 1D dynamic, diffusive, or kinematic wave equations for flows and water surface elevations over a network of conduits (i.e., pipes, canals, and rivers) and sub-catchments connected together at their endpoints by nodes (i.e., junctions, outfalls, storage units, and flow dividers) (Rossman, 2006).

We developed the SWMM HydroCouple Component from the EPA SWMM source code by modifying the code to expose those boundary data that are needed for coupling including the inflows, outflows, and water surface elevations at the inlets and outlets of the stormwater conveyance system. In developing the SWMM HydroCouple component, we employed the HydroCouple network interface definition to represent the network of SWMM nodes connected together by conduits. The benefit of this is that the inflows and outflows as well as water surface elevation data that the SWMM component supplies to components coupled to it provide the topological information that is needed to traverse the entire stormwater conveyance system.

Developing this component required knowledge of C/C++ so that we could modify the SWMM code, which was written in C and then recompile it into a compliant component. This process would be similar for other models, but, in general, the level of effort required to develop a component is subjective. It depends on the programming skill of the developer, the degree of understanding about the model being wrapped, and the complexity of the model component being developed.

The SWMM model code has been parallelized in many sections to improve performance using OpenMP as described by Burger et al. (2014). The parallelism introduced in SWMM is a shared memory type of parallelism (i.e., it uses a single MPI process but scales with number of cores allocated to the MPI process). This feature served as the basis for our investigation of the performance of the coupled model as more CPU resources were added.

We developed the SWMM model instance using detailed survey data of sizes for stormwater pipes, inlets, and outfalls provided in shapefile format by Logan City. The conduit diameters ranged from 0.30 to 1.38 m, with lengths ranging from 0.5 to 390 m. This dataset resulted in a SWMM model with 1769 conduits, 2093 junction inlets, and 138 outfalls. We executed the SWMM model using SWMM's adaptive timestep option with a minimum timestep of 0.01 s and a maximum time step of 5 s. A maximum number of 20 iterations was selected for each time-step.

4.3. FVHM model development

FVHM was developed to solve the shallow water equations over an unstructured triangular irregular network (TIN) mesh using an implicit finite volume method. Details for the formulations and hydrologic process representations used are provided in Appendix A. In the context of this case study, FVHM was developed to simply route flows in riverine and overland areas without many of the hydrologic processes that are typically represented in hydrologic models (e.g., infiltration, evapotranspiration, etc.). However, FVHM fully exposes the geospatial data structures needed to demonstrate how different configurations can be used to couple 1D and 2D models and run them on HPC. We purposefully developed FVHM for this case study to focus on the model coupling data structures and configurations without the complexity introduced by representing many hydrologic processes in the 2D model component. Because the hydrologic processes represented in the 2D model are independent of the data structures and coupling configuration, the techniques we demonstrate here are generalizable to 2D models that include more detailed hydrologic process representations. Given that our 2D model does not represent processes like infiltration, the model configuration presented here represents a worst-case scenario simulation and is a useful exercise for evaluating the performance of the stormwater infrastructure under a high intensity rainfall event.

In developing the FVHM model, we directly utilized the data structures provided as part of the HydroCouple interface definitions. For example, the computational mesh used in FVHM adopts a TIN interface definition with its associated quad-edge data structure. For creating boundary conditions, the geometry interfaces prescribed by the OGC SFA specification were adopted. For instance, polygons were used to demarcate the area where precipitation inputs apply. Polylines were used to define the mesh edge boundaries where inflows and outflow apply. The implementations of these interfaces and the file input and output implementations provided by the HydroCouple SDK simplified the model development process by allowing us to focus on the computational parts of the code we were actually interested in.

We were also able to use the capabilities of HydroCouple to enable parallelism in executing the model. FVHM uses both fine and coarse-grained parallelism in its code. Fine grained parallelism using OpenMP is employed in several areas in looping over each of the computational cells to calculate spatial gradients of water surface elevations and velocities, friction, and to apply boundary conditions. Since FVHM uses an implicit time marching scheme, systems of linear equations need to be solved at each time step to compute velocities and water surface elevations for each cell. FVHM uses the hypre software library (Falgout and Yang, 2002), which solves large, sparse linear systems of equations on massively parallel computers. FVHM partitions the system of equations it solves at every time step into smaller chunks to be solved in different MPI processes using the hypre library.

For the FVHM model instance, we developed its computational mesh using sub-meter, high resolution light detection and ranging (LIDAR) data collected in 2005. This dataset was supplemented with the 10-m elevation data from the United States Geological Survey's (USGS) 3D Elevation Program. The mesh contained 44861 cells with sizes ranging from 0.1 to 18900 m^2. This range of cell areas was the result of refining the model mesh along the canal where we were interested in evaluating in more detail and coarsening the mesh in the upstream overland areas where we were only interested in estimating runoff. For the boundary conditions, a 30-min resolution, time varying precipitation time series for the nearest rain gauge was developed using the 25-year, 24-h design storm totaling 61.2 mm and the Natural Resources Conservation Service (NRCS) Type-II (Cronshey, 1986) rainfall distribution curve. This storm is the prescribed storm for designing urban stormwater infrastructure in Logan, thus, is a useful test for the coupled model. For diversion flow from the Logan River into the Northwest Field canal, we applied the maximum legally allowable irrigation diversion of 1.351 m^3/s for the entire duration of the storm to evaluate worst case inundation conditions (e.g., an intense storm during a time where the canal was full of irrigation water). We executed the FVHM model using the adaptive time step option with a minimum timestep of 0.01 s and a maximum time step of 5 s. A maximum number of iterations of 200 was specified for the for each time step.

4.4. 1D-2D model coupling configurations

In coupling SWMM and FVHM, we adopted different coupling configurations depending on the relationship between the water levels in the coupling cell of the FVHM model, water level in the coupling inlet or outfall/outlet of the SWMM model, ground surface elevation at the coupling interface, and the invert elevations of coupling inlets or outfalls/outlets in the SWMM model (Fig. 8). For all cases, we adopted a bi-directional exchange of boundary condition data at the coupling node interfaces between the two models. This bi-directional data exchange proceeds by passing the water surface elevation from the FVHM model to the SWMM model and then passing outflows from the SWMM model to the FVHM model.

For the case where the water surface elevation Z_m of an inlet in the 1D model is less than the bottom elevation of its overlying cell in the 2D model Z_s and the water depth in the 2D cell is greater than A_{in}/w (Fig. 10a), the free weir equation (Equation (1)) was used to estimate discharge into the inlet:

$$Q = c_w w \sqrt{2g} (Z_w - Z_s)^{\frac{3}{2}} \tag{1}$$

where Q is the discharge into the inlet, c_w is the weir discharge coefficient, w is the weir crest width, g is acceleration due to gravity, Z_w is the water surface elevation in the overlying 2D cell, and A_{in} is area of the inlet opening. If the water depth in the 2D cell is greater than A_{in}/w (Fig. 10b), the orifice equation (Equation (2)) is used:

$$Q = c_O A_{in} \sqrt{2g} (Z_w - Z_s)^{\frac{1}{2}} \tag{2}$$

where c_O is the orifice discharge coefficient. The discharges are computed in the 1D model and added as sinks in the continuity equation of the 2D cell and as sources to the inlet of the 1D model.

For the case where an inlet node of the 1D model is completely submerged by the water in the 2D model (Fig. 10c), the water level in the corresponding 2D cell is set as the water level in the receiving inlet of the SWMM model. The resulting surcharge or inflow values for the inlet are then applied as a sink/source term in the continuity equation of the 2D cell.

4.5. Performance of various HPC configurations

We executed the coupled model on up to 10 HPC nodes/MPI processes to evaluate the benefits of devoting more computing resources. Fig. 9 shows maximum inundation depths for each cell computed for the entire duration of the simulation.

The results of the experiment comparing the relative speed up as the number of MPI processes are increased to the optimal, linear speed up desired are shown in Fig. 10. Despite the rudimentary parallelism introduced into the FVHM model, the results show up to a 5 times speedup with 10 HPC nodes. Different models and different coupling configurations may achieve different levels of speedup, but this demonstration illustrates the capabilities of HydroCouple to enable simulations on HPC.

5. Discussion

In our search of the literature related to OpenMI, we found numerous papers examining the mechanics for developing OpenMI components and applying them to hypothetical or real-world model coupling scenarios, assessing the performance models that are coupled using OpenMI, and the interoperability of OpenMI with other modeling frameworks, hydrologic information systems, and web-services technologies. For example, Elag et al. (2011) assessed the mass-balance errors of a hypothetical coupling scenario involving the coupling of simplified solute transport models including a surface and a sediment media model that were temporally misaligned. Bulatewicz et al. (2010) successfully coupled agricultural, groundwater, and economic models to evaluate the impacts of alternative water use policies for a major aquifer in the US. Goodall et al. (2011) and Castronova et al. (2013) demonstrated the service-oriented modeling paradigm, where remote models were coupled using web-services technologies like the OGC's Web Processing Service (WPS) protocol. We found it encouraging that OpenMI has been used for a variety of modeling use cases, which further supports our decision to do this work in the context of OpenMI.

However, we found relatively few studies that specifically focused on the architectural design of OpenMI and its advantages and drawbacks. Lloyd et al. (2011) assessed the degree of "framework invasiveness" of OpenMI and other component-based modeling frameworks using a single modeling use case across multiple frameworks. They defined "framework invasiveness" as the degree of dependency between a modeling framework implementation and its associated models. OpenMI showed a moderate degree of "framework invasiveness" when compared to OMS, which was lower, and ESMF, which was higher. However, there was some subjectivity in the implementation of the model components that were coupled across the various frameworks, and their study used the older OpenMI 1.4 Version.

Knapen et al. (2013) examined the suitability of using OpenMI as a model integration platform across disciplines through a workshop that was organized to solicit feedback from both software developers and model application specialists. In addition to identifying all the benefits of using OpenMI we enumerated above, they identified the following as areas where improvements could be made: 1) provision of data analysis and visualization tools, 2) support for multi-threaded executions, 3) programming language bridges, 4) support for standard spatiotemporal data structures, 5) semantic integration, and 6) support for models outside of the hydrologic modeling realm. It must be noted that this assessment was also conducted using the older OpenMI Version 1.4. The newer OpenMI Version 2.0 addresses some of these challenges including the provision of interfaces that make it easier to integrate semantics and support for models outside of the hydrological modeling

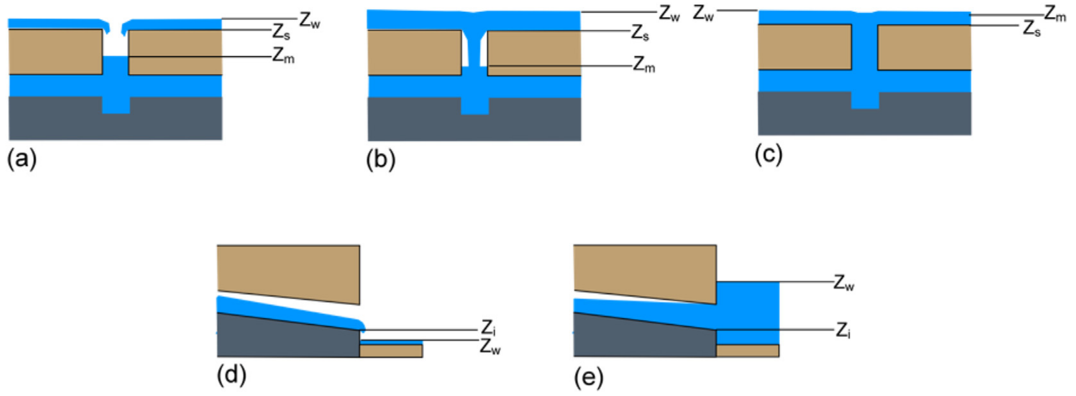

Fig. 8. 1D-2D Model interaction scenarios: (a) 1D Inlet water surface elevation (Z_m) less than 2D cell bottom elevation (Z_s) and 2D cell water depth less than A_{in}/w; (b) 1D Inlet water surface elevation (Z_m) less than 2D cell bottom elevation (Z_s) and 2D cell water depth greater than A_{in}/w; (c) 1D Inlet water surface elevation (Z_m) greater than 2D cell bottom elevation (Z_s); (d) Water surface elevation of 2D coupling cell (Z_w) less than outfall invert elevation (Z_i) of 1D model; (e) Water surface elevation of 2D cell greater than outfall invert elevation of 1D cell (Z_i).

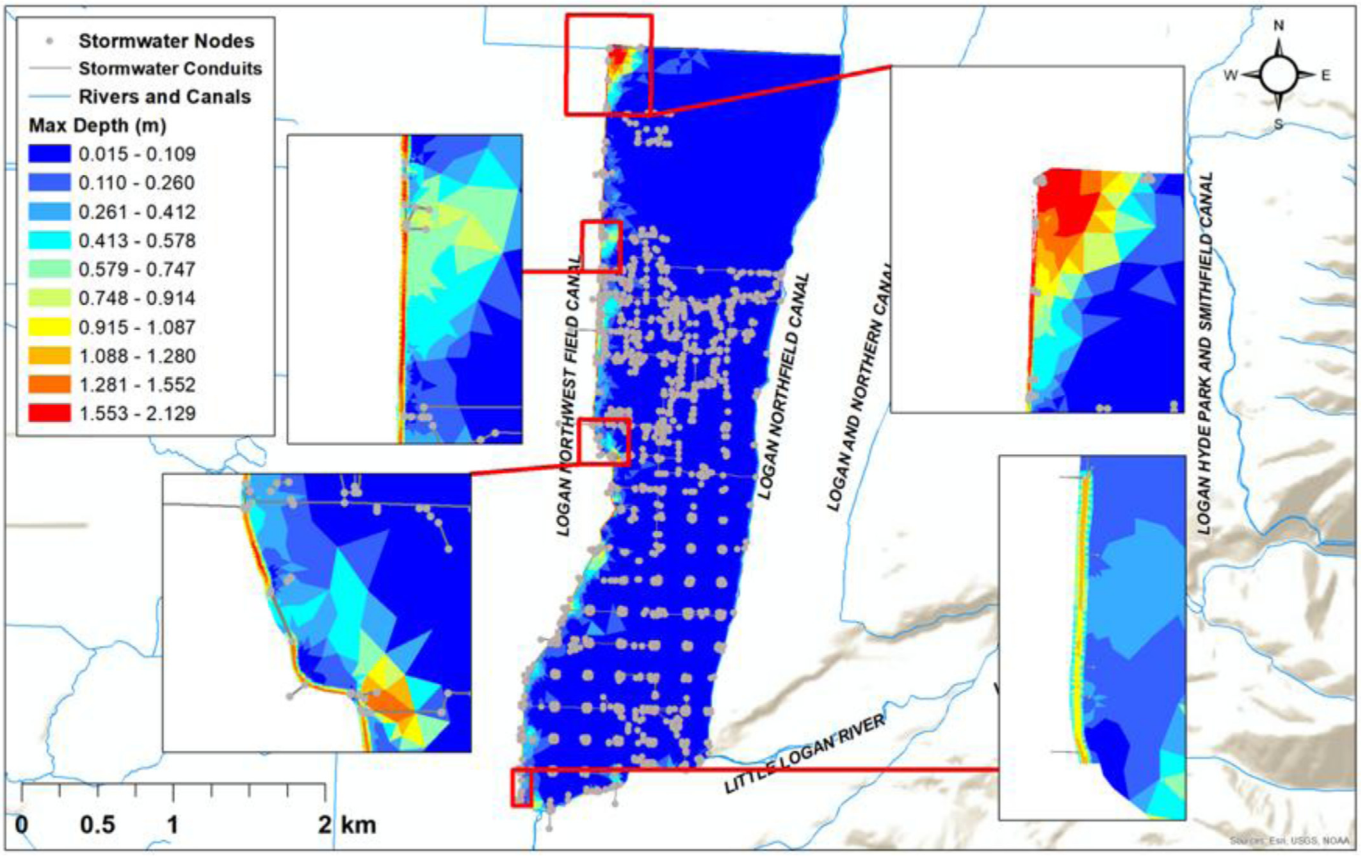

Fig. 9. Map of maximum inundation depths for the simulation period.

realm. The work presented in this paper addresses some of the remaining challenges highlighted in the workshop discussed by Knapen et al. (2013) as well as those we encountered in our use of OpenMI.

HydroCouple's interface definitions and associated software tools build on the strengths of the OpenMI 2.0 standard by advancing interface definitions to better facilitate water resources modeling applications and data structures that have heretofore been missing in existing component-based frameworks. These interfaces include new topologically aware geospatial data structures based on widely accepted standards like the OGC SFA specification, customizable data exchange workflows, and support for parallelized simulations on HPC infrastructure.

While the HydroCouple interface definitions are programming language agnostic, we implemented them using C++ to ensure that the code can be compiled on most operating systems. Another benefit of C++ is that, unlike C#, it has no dependency on a framework like.NET, which can play a role when installing model compositions on computational resources or on different operating systems. Compositions can be compiled in such a way that installation of the software involves copying

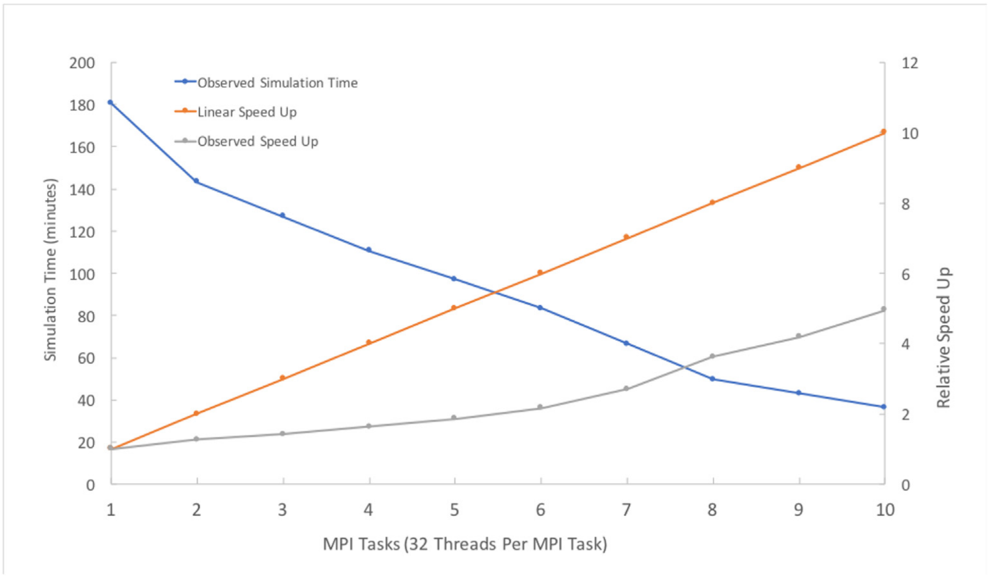

Fig. 10. Relative speed up in simulation time for increases in the number MPI processes.

the executable, dynamic link libraries, and additional files involved in a modeling application, which does not require administrative privileges. Additionally, the C++ HydroCouple implementation provides a way to avoid the data marshalling costs that often arise when legacy models are wrapped for component-based modeling frameworks using interpreted languages. C++ provides direct memory access to data for models developed using languages like C and Fortran, which have traditionally been used for model development because of their efficiency. This helps address the design considerations 1, 2, and 3 identified in Table 1 for component-based modeling frameworks. Existing OpenMI components developed by wrapping legacy models can be ported to HydroCouple components with minimal effort since HydroCouple adopts most of the OpenMI data structures and concepts. Additionally, HydroCouple provides explicit implementations of standard geospatial dataset formats and associated topological information that are missing from existing component-based modeling frameworks. These data structures are widely employed for delineating model domains and prescribing boundary conditions in many water resources modeling applications (i.e., design consideration 5 in Table 1).

In addition to better facilitating "embarrassingly parallel" experimental simulations within component-based modeling frameworks through the addition of functions that create independent copies of model instances, HydroCouple allows users to partition available HPC computing resources in a way that is transparent to model components (i.e., design consideration 4 in Table 1). The 1D-2D coupled hydraulic modeling example described illustrates the performance benefits to be gained, especially in the hydrologic modeling community by moving into the HPC arena. This application was developed to test the model code and also serve as an example of how to develop new model components for HydroCouple. The model is shared in the GitHub repository (https://github.com/hydrocouple/fvhmcomponent/examples).

The SDK and HydroCouple Composer model coupling GUI environment facilitate the development of components that use HydroCouple (i.e., design consideration 7 in Table 1). The SDK implements the core HydroCouple interfaces as well as spatiotemporal datasets and their file input and output operations so that the costs of converting existing models into components and in developing new models are reduced. The HydroCouple Composer GUI allows users to interactively select and couple models, launch simulations, monitor simulations, and visualize results of simulations on a single desktop or on HPC systems.

6. Conclusions

We advanced these new HydroCouple interface definitions in hopes that they can be considered for inclusion in future versions of the OpenMI standard. While we developed HydroCouple based on the lessons we learned using OpenMI so that it addresses the drawbacks we encountered, we understand that it may not completely support the wide array of simulation types undertaken by water resources modelers and the wider earth systems modeling community in general. While we have used the name "HydroCouple" to describe the software presented in this paper and have described it in terms of water resources modeling applications, the advancements introduced in this manuscript are also applicable to other earth systems modeling fields.

We envision the continued improvement of the interfaces prescribed through HydroCouple as a community effort and have, therefore, shared the interface definitions, HydroCouple Composer GUI, as well as all the components we have developed so far in a transparent manner in a publicly accessible source code repository (https://github.com/hydrocouple). Through this repository, users can contribute new HydroCouple components and improvements to the HydroCouple interfaces definitions, its associated software, and available components back to the repository for the benefit of the larger component-based modeling community. We have also communicated the advancements we have implemented to the OpenMI Association so that they can be considered for inclusion in future versions of the OpenMI standard.

Unlike their atmospheric modeling counterparts who have had a long history of executing their models on HPC infrastructure, most water resources modeling practitioners have traditionally executed their models on single desktop machines. There is, however, an increasing recognition that HPC is needed to tackle challenging problems such as simulating the interaction of land surface hydrologic processes with the atmosphere at the global scale, evaluation of different model structures, and uncertainty assessment. HydroCouple supports this direction by prescribing interface definitions that allow users to partition CPU and GPGPU computing resources among components for more efficient simulations.

The proliferation of many component-based modeling frameworks has made it difficult to fully achieve their promises of reusability and ability to conduct more holistic modeling evaluations. While HydroCouple does not fully address this challenge, it is not a completely new effort developed from scratch as so many other proposed component-based modeling frameworks. As much as was possible, it builds on the core concepts and interface definitions used in the widely cited OpenMI standard and, therefore, makes translating the many existing OpenMI components into HydroCouple components and vice-versa relatively inexpensive. For example, the SWMM component described in this manuscript was adapted from one originally developed for our previous OpenMI application described in Buahin and Horsburgh (2015). It is clear that in order to avoid the challenge of over proliferation of component-based modeling frameworks, framework interoperability needs to be a priority among component-based modeling practitioners. Many component-based modeling frameworks share common characteristics (e.g., the specification of IRF interfaces). These common characteristics can be used as leverage to create components that are able to use different frameworks. HydroCouple supports this goal by providing framework and programming language independent interface definitions. Furthermore, work is ongoing to create a wrapper component that automatically wraps components that implement the CSDMS BMI interface definitions so that they can be coupled to other models within the HydroCouple framework.

Acknowledgements

This research was supported by National Science Foundation EPSCoR Grant IIA 1208732 awarded to Utah State University as part of the State of Utah EPSCoR Research Infrastructure Improvement Award. Any opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Appendix A. FVHM HydroCouple Component Model Formulation

The FVHM HydroCouple model solves the shallow water equations (SWE) (Equations (A.1) and (A.2)) using the finite volume approximation over a triangular irregular network mesh:

$$\frac{\partial \zeta}{\partial t} + \nabla\cdot(h\vec{V}) = q \tag{A.1}$$

$$\frac{\partial h\vec{V}}{\partial t} + \nabla\cdot(h\vec{V}\vec{V}) = -gh\nabla\zeta - \frac{\boldsymbol{\tau_b}}{\rho} + \nabla\cdot(\Gamma h\nabla\vec{V}) + Fh \tag{A.2}$$

where ζ is the water surface elevation; t is time; h is the water depth; $\overrightarrow{V}$ is the velocity vector; q is the sum of external fluxes; g is the acceleration due to gravity; τ_b is the bed shear stress (friction) vector; ρ is water density; Γ is the sum of the kinematic (ν) and eddy viscosities (ν_t); and F is the vector sum of external forces. The bed shear stress is calculated using the Manning's roughness equation shown in Equation (3):

$$\begin{pmatrix}\tau_{bx}\\ \tau_{by}\end{pmatrix} = \frac{\rho g n^2}{\sqrt[3]{h}}\begin{pmatrix}u\\ v\end{pmatrix}\sqrt{u^2+v^2} \tag{A.3}$$

where n is the Manning's roughness coefficient and u and v are the velocities in the x and y direction respectively. The FVHM component was developed to especially handle hydrologic modeling applications that often involve prolonged periods of dry spells with non-existent or small water depths, which is necessary to simulate areas with climate comparable Utah and the intermountain western U.S.

The finite volume approximation estimates the average value of a conserved quantity in an arbitrarily shaped control volume using an integral version of partial differential shallow water equations. The theorem underlying the finite volume method is Gauss's divergence theorem (Equation (4)), which may be interpreted physically as the integral of the divergence of a vector (i.e., **a**) in a control volume (i.e., CV) is equal to the sum of the components of the vector normal (i.e., **n**) to surfaces of area of the control volume (i.e., A) (Versteeg and Malalasekera, 2007).

$$\int_{CV}(\nabla\cdot\mathbf{a})\,\mathrm{d}V = \int_A \mathbf{n}\cdot\mathbf{a}\,\mathrm{d}A \tag{A.4}$$

To illustrate how the Gauss theorem is used in the derivation of the finite volume numerical approximations for the shallow water equations in the FVHM Component, we apply it to the transport of velocity $\overrightarrow{V}$ in the control volume P surrounded by neighboring control volumes N1, N2, and N3 as depicted in Fig. A1. In Fig. A1, a, b, and c represent the nodes of the triangle for the control volume P; $\nabla\eta$, $\boldsymbol{e_\eta}$, $\boldsymbol{e_n}$, C and represent the length, unit vector, unit normal vector, and centroid of the common edge between control volume P and its neighboring control volume N1, respectively; $\nabla\xi$ and e_ξ represent the length and unit vector for the distance between the centroids of the control volumes P and N1 respectively; and $\mathbf{r}_{PC}$ and $\mathbf{r}_{CN}$ represent the vector distances between the centroids of the control volumes P and N1 with the centroid C of their common edge respectively.

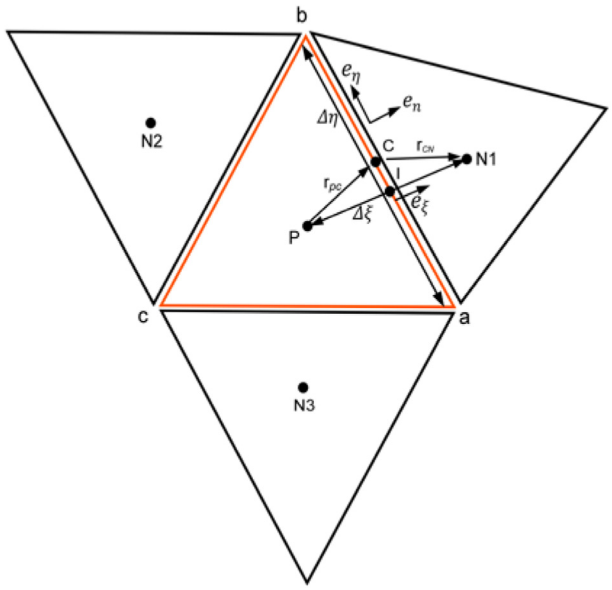

Fig. A.1. Control volume P surrounded by neighboring control volumes N1, N2, and N3.

In FVHM, the collocated grid arrangement used by several investigators including Peric (1985) and Lai (2009) is adopted. This grid arrangement involves calculating the control volume velocities, depths, and water surface elevations at the centroid of each control volume. Applying the Gauss theorem to the momentum conservation equations yields Equation (5).

$$\int_A \frac{\partial h\overrightarrow{V}}{\partial t}\mathrm{d}A + \int_\eta e_n\cdot(h\overrightarrow{V}\overrightarrow{V})\mathrm{d}\eta = \int_A(-gh\nabla\zeta)\mathrm{d}A - \int_A\left(\frac{\tau_b}{\rho}\right)\mathrm{d}A + \int_\eta e_n\cdot(\Gamma h\nabla\overrightarrow{V})\mathrm{d}\eta + \int_A F\mathrm{d}A \tag{A.5}$$

where A is the area of the control volume. The numerical approximations for the terms in Equation (5) are as follows:

$$\int_A \frac{\partial h\overrightarrow{V}}{\partial t}\mathrm{d}A = \frac{\left(h^{n+1}\overrightarrow{V}^{n+1} - h^n\overrightarrow{V}^{n}\right)A}{\Delta t} \tag{A.6}$$

$$\int_A\left(\frac{\tau_b}{\rho}\right)\mathrm{d}A = \frac{\tau_b}{\rho}A \tag{A.7}$$

$$\int_A(-gh\nabla\zeta)\mathrm{d}A = -gh\begin{pmatrix}\frac{\partial\zeta}{\partial x}\\ \frac{\partial\zeta}{\partial y}\end{pmatrix}A \tag{A.8}$$

$$\int_A Fh\mathrm{d}A = FhA \tag{A.9}$$

where the superscript n+1 and n represents the current timestep and the previous timestep respectively, and Δt is the current timestep.

The water surface elevation spatial gradients $\frac{\partial\zeta}{\partial x}$, $\frac{\partial\zeta}{\partial y}$, and all spatial gradients in FHVM are estimated using the least-squares gradient reconstruction approach from cell centered values of neighboring cells in the previous time step/previous iteration. Referring to Fig. A1, the water surface elevation of a neighboring cell N (i.e., ζ_N) surrounding cell P can be estimated from the water surface gradient at P using Equation (10):

$$\zeta_N = \zeta_P + \left.\frac{\partial\zeta}{\partial x}\right|_P (\Delta x_{PN}) + \left.\frac{\partial\zeta}{\partial y}\right|_P (\Delta y_{PN}) \tag{A.10}$$

where Δx_{PN} and Δy_{PN} represent the distances in the x and y directions from the centroid of the control volume P to the centroid of the neighboring control volume N. Assembling Equation (10) for all neighboring cells into a linear system of equations yields Equation (11), which is solved using the QR decomposition method in FVHM.

$$\begin{bmatrix} \Delta x_{PN1} & \Delta y_{PN1} \\ \Delta x_{PN2} & \Delta y_{PN2} \\ \Delta x_{PN3} & \Delta y_{PN3} \end{bmatrix} \begin{bmatrix} \left.\frac{\partial\zeta}{\partial x}\right|_P \\ \left.\frac{\partial\zeta}{\partial y}\right|_P \end{bmatrix} = \begin{bmatrix} \zeta_1 - \zeta_P \\ \zeta_2 - \zeta_P \\ \zeta_3 - \zeta_P \end{bmatrix} \tag{A.11}$$

The derivation of the numerical approximations of the more complex diffusion and advection terms in the momentum equation are provided in the following sections.

A.1. Discretization of the Diffusion Term

The diffusion term in the momentum equation is discretized as follows:

$$\int_\eta e_n\cdot(\Gamma h\nabla\vec{V})\mathrm{d}\eta = \sum_{all\ sides} \Gamma_c h_c e_n\cdot\left(\frac{\partial\vec{V}}{\partial n}e_n + \frac{\partial\vec{V}}{\partial\eta}e_\eta\right)\Delta\eta \tag{A.12}$$

where Γ_c and h_c are the viscosity and the water depth at the centroid of the common edge between the control volume P and it neighboring cell. These values are estimated using gradients calculated from the gradients derived from the least-squares gradients reconstruction method described earlier.

It can be shown from trigonometry as detailed in Versteeg and Malalasekera (2007) that the direct gradient and cross diffusion terms of Equation (A.12) for each neighboring cell can be represented by Equation (A.13):

$$e_n\cdot\left(\frac{\partial\vec{V}}{\partial n}e_n + \frac{\partial\vec{V}}{\partial\eta}e_\eta\right) = \underbrace{\frac{e_n\cdot e_n}{e_n\cdot e_\xi}\frac{\vec{V_N} - \vec{V_P}}{\Delta\xi}}_{Direct\ Gradient} + \underbrace{\frac{e_\xi\cdot e_\eta}{e_n\cdot e_\xi}\frac{\vec{V_b} - \vec{V_a}}{\Delta\eta}}_{Cross-diffusion} \tag{A.13}$$

where $\vec{V_P}$ and $\vec{V_N}$ are the cell velocities for P and N respectively, and $\vec{V_b}$ and $\vec{V_a}$ are cell P's interpolated nodal velocities for the shared edge between cells P and N.

Two approaches are available in FVHM for computing the cell turbulent eddy viscosity. The first is the parabolic eddy viscosity model (Equation (A.14)):

$$\nu_t = c_t U_* h \tag{A.14}$$

where c_t is theoretically equal to $\frac{\kappa}{6}$, with κ being the von Kármán constant (Wu et al., 2014). The second eddy viscosity model is the Smagorinsky–Lilly model (Smagorinsky, 1963) shown in Equation (A.15):

$$\nu_t = c_s A\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2} \tag{A.15}$$

where c_s is the Smagorinsky constant, with values that are usually between 0.1 and 0.2.

A.2. Discretization of the Advection Term

Following Lai (2009), the advection term in the momentum equation is discretized as follows:

$$\int_\eta e_n\cdot(h\vec{V}\vec{V})\mathrm{d}\eta = \sum_{all\ sides}(h_c V_c)^{n+1,\#}(\vec{V_c})^{n+1}\Delta\eta \tag{A.16}$$

where the superscript $n+1,\#$ refers to the previous iteration for the current timestep; V_c is the normal velocity to the current edge at its centroid, where a positive value indicates an outward flow from the control volume and a negative value indicates inflow into the control volume; and $\vec{V_c}$ is the velocity vector at the centroid of the current edge that is to be calculated at the current timestep.

$\vec{V_c}$ is estimated as a function of the velocity of the current cell P and its neighboring upstream and downstream cell velocities using Equation (A.17):

$$\vec{V_c} = \vec{V_P} + \frac{1}{2}\psi(r)(\vec{V_D} + \vec{V_P}) \tag{A.17}$$

where $\vec{V_D}$ is the velocity of the control volume downstream of the current cell P, and $\psi(r)$ is a flux limiting function of r. The variable r is the upwind ratio of consecutive gradients of velocity defined using Equation (A.18):

$$r = \frac{\vec{V_P} - \vec{V_U}}{\vec{V_D} - \vec{V_P}} \tag{A.18}$$

where $\vec{V_U}$ is the velocity of the control volume upstream of the current cell P. Equation (A.17) assumes that the centroid of the edge C is equidistant from the centroids of the bounding control volumes, hence the multiplication factor 0.5. However, for unstructured grids this may not be true. Therefore, an inverse distance interpolation weighting factor L is applied in Equation (A.19) instead of the 0.5 as recommended by (Denner and van Wachem, 2015):

$$L = \frac{r_{PC}}{r_{PC} + r_{CN}} \tag{A.19}$$

Assuming the direction of flow is from P to N1, it is easy to locate the downstream control volume with velocity $\vec{V_D}$ in Figure A1 that is to be used in calculating r. However, finding the upstream control volume $\vec{V_U}$ is not straightforward. To overcome this challenge, Darwish and Moukalled (2003) derived Equation (A.20) for calculating r:

$$r = \frac{2\nabla\vec{V_P}\cdot r_{PN}}{\vec{V_D} - \vec{V_U}} \tag{A.20}$$

where r_{PN} is the vector distance between the centroids of the control volumes P and N.

Using $\psi(r) = 0$ leads to the edge velocity being the same as the cell velocity representing the upwind differencing scheme. While the upwind differencing scheme is stable and results in smooth solutions, it is only first-order accurate. $\psi(r) = 1$ represents the central differencing scheme, which, while second-order accurate, can lead to spurious oscillations with problems that exhibit sharp discontinuities in velocities as is common with higher-order schemes. To obtain stable and non-oscillatory solutions for higher-order schemes, the function $\psi(r)$ must be monotonicity-preserving. Monotonicity-preserving schemes ensure that solutions do not create local extrema. Additionally, the value of a local minimum must be non-decreasing and the value of a local maximum must be non-increasing (Versteeg and Malalasekera, 2007). Monotonicity-preserving schemes have a property that the total variation (TV) (i.e., Equation (A.21)) of the discrete solutions should be total variation diminishing (TVD). TVD schemes are characterized with TV values that decrease with time as shown in Equation (A.22).

$$TV = \int \left|\frac{\partial u}{\partial x}\right| dx \tag{A.21}$$

$$TV\left(u^{n+1}\right) \leq TV\left(u^{n}\right) \tag{A.22}$$

Sweby (1984) provides the necessary and sufficient conditions for $\psi(r)$ to be TVD in terms of a relationship between r and $\psi(r)$. Several $\psi(r)$ functions that meet these conditions are provided in FVHM, including those shown in Table A.1.

Table A.1
TVD flux limiter functions.

Name	Limiter function $\psi(\boldsymbol{r})$	Source
Van Leer	$\frac{r+\lvert r\rvert}{1+r}$	van Leer (1974)
Van Albada	$\frac{r+r^2}{1+r^2}$	van Albada et al. (1982)
UMIST	$\max\left[0, min\left(2r, \frac{1+3r}{4}, \frac{3+r}{4}, 2\right)\right]$	Lien and Leschziner (1994)
QUICK	$\max\left[0, min\left(2r, \frac{3+r}{4}, 2\right)\right]$	Leonard (1988)
Min-Mod	$\begin{cases} \min(r, 1) & if\ r > 0 \\ 0 & if\ r \leq 0 \end{cases}$	Roe (1985)

A.3. Pressure Velocity Coupling

The discretization for the momentum equations provided can be organized into a linearized system of equations for all control volumes based on the control volume center values and can be solved implicitly for new velocities using Equation (A.23):

$$a_P\vec{V_P} = \sum_{N=1}^{3} a_N\vec{V_N} - ghA\nabla\zeta + \vec{S}hA \tag{A.23}$$

where a_P is the coefficient of the velocity of the current cell P, a_N are the coefficients of the neighboring cells, and $\vec{S}$ is the sum of the external forces and constants acting on the control volume. We seek to solve Equation (A.23) for control volume velocities as well the water surface elevations. The momentum equation is non-linear because it involves the multiplication of two velocity terms. Additionally, for incompressible flows, cell velocities and pressures are coupled in a non-linear fashion through the momentum and continuity equations. In FVHM, the Semi-Implicit Method for Pressure Linked Equations (SIMPLE; Patankar and Spalding, 1972) or alternatively SIMPLE-Consistent (SIMPLEC; Doormaal and Raithby, 1984) iterative solution procedures are adopted to deal with these nonlinearities.

First, it is important to note that in the collocated grid arrangement, where velocities and water surface elevation values are estimated for the centroid of each control volume, a highly non-uniform water surface elevation field can act like a uniform field when the gradients of the water surface elevation fields are calculated numerically. This may lead to the well-known "checker-board" pressure field effect, which, in turn, leads to non-physical results (Versteeg and Malalasekera, 2007). To overcome this problem, Rhie and Chow (1983) proposed the momentum interpolation equation shown in Equation (A.24) to calculate the edge normal velocity V_c:

$$V_c = ((1.0 - L)\overrightarrow{V_P} + (L)\overrightarrow{V_N})\cdot e_n - \frac{1.0}{r_\xi \cdot e_n}\left(\frac{(1.0 - L)A_P}{a_P} + \frac{(L)A_N}{a_N}\right)(gh_c(\zeta_N \; - \; \zeta_P) + (gh_P(1.0 - L)(\nabla\zeta)_P + gh_N(L)(\nabla\zeta)_N)\cdot r_\xi) \tag{A.24}$$

The SIMPLE and SIMPLEC iterative solution procedure begins by using the initial or previous iteration water surface elevation values ζ^* to solve the momentum equation (i.e., Equation (A.23)), for an intermediate velocity field $\overrightarrow{V}^*$ as shown in Equation (A.25).

$$a_P \overrightarrow{V}_p^* = \sum_{N=1}^{3} a_N \overrightarrow{V}_N^* - ghA\nabla\zeta^* + \overrightarrow{S} \tag{A.25}$$

Since the initial water surface elevation used is a guess from the previous iteration or time step, the computed velocities are likely not correct. A water surface elevation ζ' that corrects the water surface elevation ζ^* is, therefore, defined as shown in Equation (A.26). Similarly, a new velocity $\overrightarrow{V}'$ that corrects the calculated intermediate velocity $\overrightarrow{V}^*$ is also defined.

$$\zeta_P = \zeta_*^P + \zeta_{'}^P \tag{A.26}$$

$$\overrightarrow{V_P} = \overrightarrow{V_*}^P + \overrightarrow{V_{'}}^P \tag{A.27}$$

Subtracting Equation (A.25) from 23 yields:

$$a_P \overrightarrow{V}_p' = \sum_{N=1}^{3} a_N \overrightarrow{V}_N' - ghA\nabla\zeta' \tag{A.28}$$

The velocity correction is then calculated from Equation (A.28) as:

$$\overrightarrow{V}_p' = \frac{\sum_{N=1}^{3} a_N \overrightarrow{V}_N' - gh\nabla\zeta'}{a_P} \tag{A.29}$$

Ignoring the minor terms in Equation (A.29) and inserting it into Equation (A.27) yields:

$$\overrightarrow{V_P} = \overrightarrow{V_*}^P + \frac{-ghA\nabla\zeta'}{a_P} \tag{A.30}$$

The velocities estimated in Equation (A.25) do not satisfy the continuity equation. Therefore, the water surface elevation correction equation (i.e., Equation (A.26)) and the correction velocity equation (i.e., Equation (A.30)) are used in the continuity equation to derive the water surface elevation correction values ζ'. These computations form the basis for the SIMPLE method. To illustrate it, we derive the finite volume approximation of the continuity equation as:

$$\left(\frac{\zeta_p^{n+1} - \zeta_p^n}{\Delta t} A_P\right) = \sum_{N=1}^{3} \Delta\eta h_c V_c + qA_P \tag{A.31}$$

where q represents external inflows. Inserting the water surface elevation correction equation and the Rhie-Chow interpolated control volume edge velocities from the velocity correction equation into the continuity equation yields:

$$\left(\frac{\left(\zeta_p^* + \zeta_p'\right)^{n+1} - \zeta_p^n}{\Delta t} A_P\right) = \sum_{N=1}^{3}\left[\Delta\eta \; h_c^{n+1}\left(V_c - \frac{1.0}{r_\xi \cdot e_n}\left(\frac{(1.0 - L)A_P}{a_P} + \frac{(L)A_N}{a_N}\right)gh_c\left(\zeta_N' - \zeta_p'\right)\right)^{n+1}\right] + qA_P \tag{A.32}$$

In the SIMPLEC method, the minor terms ignored in Equation (A.30) are included in the continuity equation to estimate that water surface elevation correction values.

To recap, for each time step, the solution process begins by using the initial or previous iteration values of water surface elevations and velocities to calculate intermediate velocity values $\overrightarrow{V}^*$ for each control volume. The water surface correction equation (Equation (A.32)) is then solved to obtain correction values ζ' for each control volume. The water surface values and velocities are then corrected using Equations 26 and 27, respectively. If convergence is not achieved, the whole process is repeated.

All the systems of equations generated by FVHM are solved using the algebraic, multigrid, preconditioned, generalized, minimal residual method (GMRES) from the hypre software library (Falgout and Yang, 2002), which solves large, sparse linear systems of equations on massively parallel computers. Even though the Courant-Friedrichs-Lewy (CFL) condition for stability does not apply for the implicit time-marching approach adopted in FVHM, the use of an excessively large time step can lead to inaccurate estimates (Durran, 2013). An adaptive time step approach was, therefore, adopted using a user specified maximum Courant number as a controlling variable. The time step at the beginning of each iteration is estimated by dividing the maximum user specified Courant number (C_o) by the maximum Courant number for the control volumes at the current time step as shown in Equation (A.33).

$$\Delta t = \frac{C_o}{\left(\sum \frac{\textit{Outgoing fluxes}}{\textit{Volume}}\right)_{max}} \tag{A.33}$$

A.4. Wetting and Drying

Tracking of the wetting front at the boundaries between wet and dry cells in hydraulic models is important because of the numerical instabilities that would arise from the unrealistically high velocities that would be calculated by dividing volumetric fluxes by the small water depths in dry cells (Kim et al., 2012). Many approaches have been proposed for the proper treatment of wetting and drying cells in hydraulic models, including the thin film, element removal, depth extrapolation, and negative depth algorithms as discussed by (Medeiros and Hagen, 2013). A common feature for the treatment of wetting and drying cells in many hydraulic models involves first classifying cells as wet, partially wetted, or dry depending on the number of cell nodes that are submerged. A wet cell has a water surface elevation that submerges all the nodes of the cell by a certain small threshold value (e.g., 1e-7 m). A partially wetted cell has a water surface elevation that submerges at least one node of a cell by a certain small threshold value. A dry cell has a water surface elevation that does not submerge any of the nodes of a cell. For each time step, the momentum equations are only solved for wet cells, and velocities for dry cells are set to zero. Velocities and water surface elevations for partially submerged cells are then extrapolated from neighboring wet cells.

The application of this approach to hydrological simulations that solve the full dynamic wave model, however, introduces some challenges. Hydrological simulations often involve long periods with small or no runoff generation. Additionally, hydrological models often involve large areas. Coarse and often steeply sloped computational cells are often employed for computational efficiency. This leads to the frequent occurrence of partially wetted cells that cause a no-flow phenomenon where water is unable to leave a cell because of ponding in the lowest corner of a cell where the water surface elevation is below the two nearest cell edge midpoints (Begnudelli et al., 2006; Kim et al., 2012; Warnock et al., 2014). Also, the typical assumption made in many hydraulic models that the water surface elevation at the centroid is equal to the water depth plus the elevation of the centroid of the cell does not hold for partially wetted cells. To address this challenge, we adopted the volume-free surface relationship (VFR) proposed by Begnudelli and Sanders (2006) in FVHM to deal with partially wetted cells. The VFR relationship makes a distinction between the free water surface elevation at the centroid and the depth at the centroid by assuming sheet-flow for partially wetted cells. This is done by calculating the flow depth as a ratio between the fluid volume in the cell and the area of the cell. The VFR approach provides equations to quickly transform water surface elevation to depths and vice versa for triangular cells to support modeling.

In FVHM, the momentum equations are solved for wet cells and partially wetted cells with depths above a specified threshold while setting velocities for dry cells to zero. The mass balance equations are then solved for all cells in the model domain.

A.5. FVHM Component Verification

The FVHM component was verified using test problems 1 and 6 from (MacDonald, 1996). Problem 1 involves subcritical flow through a channel with a rectangular cross-section. Problem 6 involves flow through a channel with a rectangular cross-section where there is a transition from subcritical flow to critical flow and then a hydraulic jump (i.e., transcritical) near the end of a channel. The attributes for the two test problems are shown in Table A.2.

Table A.2
Properties for (MacDonald, 1996) test problems.

Problem	Channel Width (m)	Channel Length (m)	Manning's Roughness (n)	Inlet Flow (m^3/s)	Outlet Water Surface Elevation (m)
1	10	1000	0.03	20	0.800054
6	10	150	0.03	20	1.700225

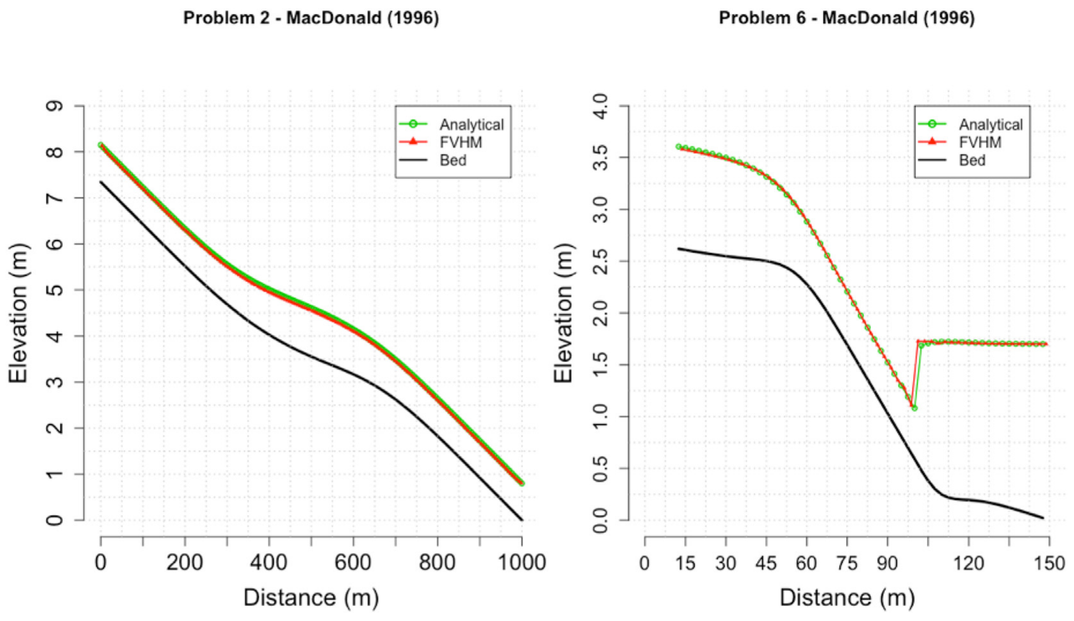

Fig. A.2. Comparison of analytical results from MacDonald (1996) test problems with FVHM.

References

van Albada, G.D., van Leer, B., Roberts Jr., W.W., 1982. A comparative study of computational methods in cosmic gas dynamics. Astron. Astrophys. 108, 76–84.

Allan, B.A., Armstrong, R.C., Wolfe, A.P., Ray, J., Bernholdt, D.E., Kohl, J.A., 2002. The CCA core specification in a distributed memory SPMD framework. Concurrency Comput. Pract. Ex. 14, 323–345.

Argent, R.M., 2004. An overview of model integration for environmental applications—components, frameworks and semantics. Environ. Model. Software 19, 219–234.

Argent, R.M., Grayson, R.B., Ewing, S.A., 1999. Integrated models for environmental management: Issues of process and design. Environ. Int. 25, 693–699.

Begnudelli, L., Sanders, B.F., 2006. Unstructured grid finite-volume algorithm for shallow-water flow and scalar transport with wetting and drying. J. Hydraul. Eng. 132, 371–384.

Beven, K.J., Warren, R., Zaoui, J., 1980. SHE: tTowards a methodology for physically-based distributed forecasting in hydrology. In: Proceedings of the Oxford Symposium 129. IAHS Publ., pp. 133–137.

Blasone, R.-S., Vrugt, J.A., Madsen, H., Rosbjerg, D., Robinson, B.A., Zyvoloski, G.A., 2008. Generalized likelihood uncertainty estimation (GLUE) using adaptive Markov chain Monte Carlo sampling. Adv. Water Resour. 31, 630–648.

Buahin, C.A., Horsburgh, J.S., 2015. Evaluating the simulation times and mass balance errors of component-based models: An application of OpenMI 2.0 to an urban stormwater system. Environ. Model. Software 72, 92–109.

Buahin, C., Horsburgh, J., 2016. From OpenMI to HydroCouple: Advancing OpenMI to support experimental simulations and standard geospatial datasets. In: International Congress on Environmental Modelling and Software, . http://scholarsarchive.byu.edu/iemssconference/2016/Stream-A/11 Accessed 17 October 2016.

Bulatewicz, T., Yang, X., Peterson, J.M., Staggenborg, S., Welch, S.M., Steward, D.R., 2010. Accessible integration of agriculture, groundwater, and economic models using the open modeling interface (OpenMI): Methodology and initial results. Hydrol. Earth Syst. Sci. 14, 521–534.

Burger, G., Sitzenfrei, R., Kleidorfer, M., Rauch, W., 2014. Parallel flow routing in SWMM 5. Environ. Model. Software 53, 27–34.

Buschmann, F., 1996. Pattern-oriented Software Architecture Volume 1: A System of Patterns. Wiley, Chichester , New York ISBN: 978-0-471-95869-7.

Castronova, A.M., Goodall, J.L., 2009. Comparing tightly coupled and loosely coupled paradigms for modeling hydrologic systems. AGU Fall Meeting Abstracts 11, 1043.

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. Environ. Model. Software 41, 72–83.

Chowdhury, R.K., Eslamian, S., 2015. Climate Change and Hydrologic Modeling. Handbook of Engineering Hydrology: Modeling, Climate Change, and Variability. CRC Press ISBN: 978-1-4665-5246-3.

Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J.E., Gutmann, E.D., Wood, A.W., Brekke, L.D., Arnold, J.R., Gochis, D.J., Rasmussen, R.M., 2015. A unified approach for process-based hydrologic modeling: 1. Modeling concept: A unified approach for process-based hydrologic modeling. Water Resour. Res. 51, 2498–2514.

Collins, N., Theurich, G., Deluca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang, W., Hill, C., Da Silva, A., 2005. Design and implementation of components in the Earth System Modeling Framework. Int. J. High Perform. Comput. Appl. 19, 341–350.

Cronshey, R., 1986. Urban Hydrology for Small Watersheds. US Dept. of Agriculture, Soil Conservation Service, Engineering Division. http://repositories.tdl.org/tamug-ir/handle/1969.3/24438, Accessed date: 27 September 2014.

Darwish, M.S., Moukalled, F., 2003. TVD schemes for unstructured grids. Int. J. Heat Mass Tran. 46, 599–611.

David, O., Ascough, J.C., Lloyd, W., Green, T.R., Rojas, K.W., Leavesley, G.H., Ahuja, L.R., 2013. A software engineering perspective on environmental modeling framework design: The Object Modeling System. Environ. Model. Software 39, 201–213.

David, O., Markstrom, S., Rojas, K., Ahuja, L., Schneider, I., 2002. The Object Modeling System. In: Ahuja, L., Ma, L., Howell, T. (Eds.), Agricultural System Models in Field Research and Technology Transfer. CRC Press.. http://www.crcnetbase.com/doi/abs/10.1201/9781420032413.ch15, Accessed date: 31 March 2014.

Denner, F., van Wachem, B.G.M., 2015. TVD differencing on three-dimensional unstructured meshes with monotonicity-preserving correction of mesh Skewness. J. Comput. Phys. 298, 466–479.

Deursen, A. van, Financial, A., 1997. Domain-specific Languages versus Object-oriented Frameworks: A Financial Engineering Case Study.

Deursen, A.V., Klint, P., Visser, J., 2000. Domain-specific languages: An annotated bibliography. Acm Sigplan Notices 35, 26–36.

Doormaal, J.P.V., Raithby, G.D., 1984. Enhancements of the simple method for predicting incompressible fluid flows. Numer. Heat Tran. 7, 147–163.

Durran, D.R., 2013. Numerical Methods for Wave Equations in Geophysical Fluid Dynamics. Springer Science & Business Media ISBN: 978-1-4757-3081-4.

Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Caron, J., Signell, R., Bentley, P., Rappa, G., Höck, H., Pamment, A., Juckes, M., 2011. NetCDF Climate and Forecast (CF) Metadata Conventions Version 1.6.

Elag, M.M., Goodall, J.L., Castronova, A.M., 2011. Feedback loops and temporal misalignment in component-based hydrologic modeling: component-based hydrologic modeling. Water Resour. Res. 47. https://doi.org/10.1029/2011WR010792.

Falgout, R.D., Yang, U.M., 2002. Hypre: a library of high performance preconditioners. In: Sloot, P.M.A., Hoekstra, A.G., Tan, C.J.K., Dongarra, J.J. (Eds.), Computational Science — ICCS 2002,Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 632–641.

Fröhlich, P.H., Franz, M., 1999. Component-oriented Programming in Object-oriented Languages. Department of, Information and Computer Science, University of California, Irvine Technical Report No. 99-49.

Goodall, J.L., Robinson, B.F., Castronova, A.M., 2011. Modeling water resource systems using a service-oriented CompAccesseduting paradigm. Environ. Model. Software 26, 573–582.

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: Open Modelling Interface. J. Hydroinf. 9, 175.

Guibas, L., Stolfi, J., 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi. ACM Trans. Graph. 4, 74–123.

Herring, J.R., 2011. OpenGIS® Implementation Standard for Geographic Information - Simple Feature Access - Part 1: Common Architecture.

Hill, C., DeLuca, C., Suarez, M., Da Silva, A., 2004. The architecture of the Earth System Modeling Framework. Comput. Sci. Eng. 6, 18–28.

Hunter, N.M., Bates, P.D., Horritt, M.S., Wilson, M.D., 2007. Simple spatially-distributed models for predicting flood inundation: A review. Geomorphology 90, 208–225.

Janssen, S., Athanasiadis, I.N., Bezlepkina, I., Knapen, R., Li, H., Domínguez, I.P., Rizzoli, A.E., van Ittersum, M.K., 2011. Linking models for assessing agricultural land use change. Comput. Electron. Agric. 76, 148–160.

Kim, J., Warnock, A., Ivanov, V.Y., Katopodes, N.D., 2012. Coupled modeling of hydrologic and hydrodynamic processes including overland and channel flow. Adv. Water Resour. 37, 104–126.

Knapen, R., Janssen, S., Roosenschoon, O., Verweij, P., de Winter, W., Uiterwijk, M., Wien, J.-E., 2013. Evaluating OpenMI as a model integration platform across disciplines. Environ. Model. Software 39, 274–282.

Komma, J., Reszler, C., Blöschl, G., Haiden, T., 2007. Ensemble prediction of floods – catchment non-linearity and forecast probabilities. Nat. Hazards Earth Syst. Sci. 7, 431–444.

Lai, Y.G., 2009. Two-dimensional depth-averaged flow modeling with an unstructured hybrid mesh. J. Hydraul. Eng. 136, 12–23.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., 2013. Integrated environmental modeling: A vision and roadmap for the future. Environ. Model. Software 39, 3–23.

Leavesley, G.H., Markstrom, S.L., Restrepo, P.J., Viger, R.J., 2002. A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling. Hydrol. Process. 16, 173–187.

van Leer, B., 1974. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. J. Comput. Phys. 14, 361–370.

Leonard, B.P., 1988. Simple high-accuracy resolution program for convective modelling of discontinuities. Int. J. Numer. Meth. Fluid. 8, 1291–1318.

Lien, F.S., Leschziner, M.A., 1994. Upstream monotonic interpolation for scalar transport with application to complex turbulent flows. Int. J. Numer. Meth. Fluid. 19, 527–548.

Lloyd, W., David, O., Ascough II, J.C., Rojas, K.W., Carlson, J.R., Leavesley, G.H., Krause, P., Green, T.R., Ahuja, L.R., 2011. Environmental modeling framework invasiveness: Analysis and implications. Environ. Model. Software 26, 1240–1250.

MacDonald, I., 1996. Analysis and Computation of Steady Open Channel Flow. University of Reading, Reading, UK.

Medeiros, S.C., Hagen, S.C., 2013. Review of wetting and drying algorithms for numerical cidal flow models. Int. J. Numer. Meth. Fluid. 71, 473–487.

Moore, R. (Ed.), 2010. The OpenMI Document Series: OpenMI Standard 2 Specification.

Moore, R.V., Tindall, C.I., 2005. An overview of the open codelling interface and environment (the OpenMI). Environ. Sci. Pol. 8, 279–286.

Parson, E.A., 1995. Integrated assessment and environmental policy making: in pursuit of usefulness. Energy Pol. 23, 463–475.

Patankar, S.V., Spalding, D.B., 1972. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. Int. J. Heat Mass Tran. 15, 1787–1806.

Peckham, S.D., Hutton, E.W.H., Norris, B., 2013. A component-based approach to integrated modeling in the oeosciences: the design of CSDMS. Comput. Geosci. 53, 3–12.

Peric, M., 1985. A Finite Volume Method for the Prediction of Three-dimensional Fluid Flow in Complex Ducts. Imperial College London (University of London).

Rhie, C.M., Chow, W.L., 1983. Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA J. 21, 1525–1532.

Roe, P.L., 1985. Some Contributions to the Modelling of Discontinuous Flows. pp. 163–193.

Rossman, L.A., 2006. Storm Water Management Model, Quality Assurance Report: Dynamic Wave Flow Routing. US Environmental Protection Agency, Office of Research and Development, National Research Management Research Laboratory.

Samuels, P.G., 1990. Cross-section Location in 1-D Models. International Conference on River Flood Hydraulics. Hydraulics Research Limited by Wiley, pp. 339–350.

Smagorinsky, J., 1963. General circulation experiments with the primitive equations. Mon. Weather Rev. 91, 99–164.

Smolders, S., Neyskens, I., Willems, P., Vaes, G., Van Assel, J., 2008. Bi-directional Sewer-river Linking through the OpenMI Software. http://web.sbe.hw.ac.uk/staffprofiles/bdgsa/11th_International_Conference_on_Urban_Drainage_CD/ICUD08/pdfs/410.pdf, Accessed date: 1 August 2014.

Sweby, P., 1984. High resolution schemes using flux limiters for gyperbolic conservation laws. SIAM J. Numer. Anal. 21, 995–1011.

Syvitski, J.P., Hutton, E.W., Peckham, S., Slingerland, R.L., 2011. CSDMS—A modeling system to aid sedimentary research. Sediment. Rec. 9, 4–9.

Szyperski, C., 2002. Component Software: Beyond Object-oriented Programming. ACM Press ISBN: 978-0-201-74572-6.

Taboada, G.L., Ramos, S., Expósito, R.R., Touriño, J., Doallo, R., 2013. Java in the high performance computing arena: research, practice and experience. Sci. Comput. Program. 78, 425–444.

Vanecek, S., Moore, R., 2014. OGC® Open Modelling Interface Interface Standard. https://portal.opengeospatial.org/files/?artifact_id=59022 Accessed 2 June 2015.

Versteeg, H.K., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics: the Finite Volume Method. Pearson Education Ltd., Harlow, England; New York ISBN: 978-0-13-127498-3 0-13-127498-8.

Voinov, A., Shugart, H.H., 2013. 'Integronsters', integral and integrated modeling. Environ. Model. Software 39, 149–158.

Warmerdam, F., 2008. The geospatial data abstraction library. In: Hall, G.B., Leahy, M. (Eds.), Open Source Approaches in Spatial Data Handling. Springer Berlin Heidelberg, pp. 87–104.

Warnock, A., Kim, J., Ivanov, V., Katopodes, N.D., 2014. Self-adaptive kinematic-dynamic model for overland flow. J. Hydraul. Eng. 140, 169–181.

Wu, W., Wang, P., Chiba, N., 2014. Comparison of five depth-averaged 2-d turbulence models for River flows. Arch. Hydro-Eng. Environ. Mech. 51, 183–200.

Zhang, G., 2009. Development of a Multi-objective Optimization Framework for Implementing Low Impact Development Scenarios in an Urbanizing Watershed. The Pennsylvania State University, State College, Pa.. https://etda.libraries.psu.edu/catalog/10187, Accessed date: 19 April 2018.