This is an Accepted Manuscript of an article published in Water Resources Research in 2011, available online: doi:10.1029/2011WR010792

Feedback loops and temporal misalignment in component-based hydrologic modeling

Mostafa Elag[1], Jonathan L. Goodall[1], and Anthony M. Castronova[1]

Mostafa Elag, Graduate Research Assistant, Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA (elag@email.sc.edu)

Jonathan L. Goodall, Assistant Professor, Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA (goodall@cec.sc.edu)

Anthony M. Castronova, Graduate Research Assistant, Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, USA (castrona@email.sc.edu)

[1]Department of Civil and Environmental
Engineering, University of South Carolina,
Columbia, South Carolina, USA.

Abstract. In component-based modeling, a complex system is represented as a series of loosely-integrated components with defined interfaces and data exchanges that allow the components to be coupled together through shared boundary conditions. Although the component-based paradigm is commonly used in software engineering, it has only recently been applied for modeling hydrologic and earth systems. As a result, research is needed to test and verify the applicability of the approach for modeling hydrologic systems. The objective of this work was therefore to investigate two aspects of using a component-based software architecture for hydrologic modeling: (1) simulation of feedback loops between components that share a boundary condition and (2) data transfers between temporally misaligned model components. We investigated these topics using a simple case study where diffusion of mass is modeled across a water-sediment interface. We simulated the multi-media system using two model components, one for the water and one for the sediment, coupled using the Open Modeling Interface (OpenMI) standard. The results were compared with a more conventional numerical approach for solving the system where the domain is represented by a single multidimensional array. Results showed that the component-based approach was able to produce the same results obtained with the more conventional numerical approach. When the two components were temporally misaligned, we explored the use of different interpolation schemes to minimize mass balance error within the coupled system. The outcome of this work provides evidence that component-based modeling can be used to simulate complicated feedback loops between

systems and guidance as to how different interpolation schemes minimize mass balance error introduced when components are temporally misaligned.

1. Introduction

Watershed-scale hydrologic models are commonly used tools in water resource management. They are applied to better understand water quality conditions, surface water and groundwater allocations, floodplain management, flood warning, and a range of water resource management activities [*Wurbs*, 1998]. Modeling hydrologic processes at the watershed-scale is challenging for many reasons including the multidisciplinary nature of watersheds where human, ecological, hydrological, and economic factors can influence how water is transported through natural and built systems. Watershed-scale modeling also introduces significant data collection and management challenges that must be addressed using sophisticated information systems and data management approaches [*Dozier*, 1992; *Horsburgh et al.*, 2008; *Goodall and Maidment*, 2009]. At an even more fundamental level, watershed-scale models must simulate hydrologic processes and flow paths that are difficult to describe and to parameterize within a simulation model (see *Beven* [2002] for a summary).

Despite these challenges, there are a number of examples of watershed-scale hydrologic models developed by the scientific and engineering community [*Singh*, 2006]. Each model has been designed to address specific requirements, yet these models include significant overlap in terms of repetitive process representations and duplication of effort. As the demands and stresses placed on water resources continues to increase, the hydrologic science community would benefit from consolidating model development effort so that a community of modelers can work collaboratively to build models capable of addressing a variety of complex water resource management challenges. We argue, along with others

[*Voinov et al.*, 2010], that a key step along this path is to understand how software engineering and information technology can be leveraged in order to allow for multi-developer modeling efforts. That is, for future watershed models to be able to keep pace with policy and decision makers' needs, there is a need to encourage community-modeling practices; and to encourage community-modeling practices, there is a need to advance the architectural approaches that underlie watershed model codes. Therefore, this work is primarily focused on the challenge of designing watershed-scale hydrologic models from a software architecture perspective.

Many examples exist of past efforts to infuse information technology into watershed modeling and management. Most of these efforts, however, have resulted in tools that target the time-intensive process of creating model input files and analyzing model output files [*Srinivasan and Arnold*, 1994], often through the integration of Geographic Information Systems (GIS) with simulation models [*Pullar and Springer*, 2000; *Miller et al.*, 2004]. While this work has resulted in important and necessary tools, it does not fully capture the potential of modern information technologies to advance how we design core architectures of watershed modeling systems. More recent efforts in the hydrologic and earth sciences have resulted in promising approaches for integrating both the growing set of hydrologic data and models [*Goodall et al.*, 2008; *Horsburgh et al.*, 2008; *Maidment*, 2008; *Voinov et al.*, 2010]. The focus of this work is on one of the more recent paradigms for structuring earth system models: component-based software architectures [*Armstrong et al.*, 2002; *Gössler and Sifakis*, 2003; *Heineman and Councill*, 2001].

Component-based software development is a method of software construction whereby a system is assembled from a set of prefabricated, reusable, and independently evolving

units of code [*Clements*, 1995]. Developing using components allows for the construction of reconfigurable software systems in a timely and more manageable manner, where components of the system can be more easily tested and validated in isolation of and in combination with other system components [*Clements*, 1995; *Garlan et al.*, 1995; *Szyperski*, 2002]. When the approach is applied for constructing modeling systems, simulations of a particular system can be constructed by coupling reusable components into a composition tailored for a specific modeling objective [*Voinov et al.*, 2010]. By adhering to component interface standards, multiple developers can contribute components to the modeling system that are able to be coupled for the simulation.

An important distinction of this work compared to previous approaches in watershed-scale hydrologic modeling frameworks is between modular and component-based approaches as the underlying architecture (Figure 1). In a purely modular approach, model integration is achieved by placing restrictions on data structures and conventions used within modular modeling routines. In contrast, component-based approaches achieve integration by standardizing the communication interface between modeling components during a simulation run [*Szyperski*, 2002]. The result is a more flexible and extensible system because it is less restrictive on the internal implementation of components. While there are many examples of technologies that emphasize a modular modeling approach for water systems (e.g., *Rahman* [2003] and *Leavesley et al.* [1996]), there are fewer examples that use a component-based approach, primarily because it is a newer approach for model integration. The Open Modeling Interface (OpenMI) [*Tindall*, 2005], however, is one example of this approach developed and used within the water resource community, and the Community Surface Dynamics Modeling System (CSDMS) [*Syvitski et al.*, 2011]

is a second example of this approach developed and used in the Earth surface dynamics modeling community. Here we focus on the OpenMI because it is meant primarily as a standard for component-based modeling in the water resources domain.

The Open Modeling interface (OpenMI) is a component interface standard developed through the European Union Water Framework Directive [*Tindall*, 2005]. Its primary purpose is to facilitate interoperability between environmental models [*Gregersen et al.*, 2007]. The basic approach used by OpenMI to couple models is to allow access to input and output values of the model directly at run-time. A model that implements an OpenMI standard interface becomes a linkable component and then can be coupled to other components through input and output exchange items [*Tindall*, 2005]. OpenMI is a pull-based pipe and filter architecture that consists of Linkable components (source components and target components) which exchange memory-based data in a pre-defined way and in a pre-defined format. The OpenMI defines the component interfaces as well as how the data is being exchanged. OpenMI version 1.4 communication protocol consists of three fundamental concepts: a linkable component, an exchange item, and a link [*Sinding et al.*, 2005]. A link connects the source and target component, and define the actual data exchange quantity between linkable components. Data exchange in an OpenMI 1.4 component composition is initiated by a trigger component that begins the data communication process. Once the composition has been triggered, components exchange data autonomously without the need for a controller to supervise the interactions. OpenMI follows a single-threaded architecture meaning that a component can handle just one request at a time. Figure 2 illustrates how models can be linked by either a uni-directional or bi-directional link in an OpenMI component composition. Uni-directional links are used

to integrate sequential models where model A is dependent on the output from model B, which itself is dependent on the output from model C. In bi-directional links, two components are linked such that each requires input from the other in order to run. This type of feedback loop communication is more complicated to implement in component-based modeling and therefore is a focus of this paper.

OpenMI version 2.0, which was released after the initiation of this study, removes the concept of a link being a distinct object and instead components directly provide their output exchange item(s) as input exchange item(s) to one or more components. In both the 1.4 and 2.0 versions of OpenMI, the target component pulls the data when needed using the pull-driven mechanism. We used the OpenMI 1.4 since the Software Development Kit (SDK) for OpenMI 2.0 is still incomplete. Our findings will remain applicable after migrating to OpenMI 2.0 because the pull mechanism with request and return of values is unchanged and so there will still be a need to model systems that have feedback loops and may require rescaling of temporally or spatially misaligned data transfers.

Past work using OpenMI for hydrologic and water resource modeling has primarily focused on applying the standard for modeling integrated water resource systems. For example, OpenMI was used to couple separate models that simulate groundwater hydrogeology, econometric farm level crop choices, and irrigated water use [*Steward et al.*, 2009; *Bulatewicz et al.*, 2010]. *Steward et al.* [2009] emphasized the advantage of component-based modeling to assembly three different components from three different disciplines. In a related application, *Fotopoulos et al.* [2010] used OpenMI to build a flood model finding that OpenMI eased the integration across various software components and increased the flexibility and extensibility of the overall modeling system. *Ewert et al.* [2009] presented

integrated assessments for policy support in agriculture that used a modeling framework built on OpenMI. The authors found that, while OpenMI provided an improved means for linking models to each other, to a GIS, or to a Decision Support Systems (DSS), the scientific basis for linking models across disciplinary boundaries and spatiotemporal scales required additional research. OpenMI has also been applied to urban hydrology to couple sewer system models with river hydraulic models, illustrating the benefit of component-based modeling which allows existing models to be coupled through the runtime exchange of data [*Reussner et al.*, 2009]. These past efforts all point to the benefits of OpenMI and component-based modeling in general for coupling disparate modeling and information systems, however, as noted by *Ewert et al.* [2009], there are important scientific questions regarding model coupling that have not been well addressed.

We suggest that two of the most pressing questions in applying component-based modeling concepts to hydrologic systems are (1) how to simulate fully coupled processes with a shared boundary condition as two distinct components with a bidirectional link and (2) how to couple two model components that have spatially and/or temporally misaligned inputs and outputs. This is not to say these are the only two questions raised in regard to component-based modeling [*Voinov and Cerco.*, 2010], but given that hydrology is complicated with many examples of feedbacks between processes operating within watershed system, a successful integrated modeling approach must be capable of simulating such interactions. Likewise, it is well known that hydrologic processes have characteristic time or length scales and that these scales can vary for different fluxes within the hydrological cycle [*Bloschl and Sivapalan*, 1995]. One of the challenges that component-based modeling attempts to address is that different components of the system can operate on indepen-

dent spatial or temporal discretizations. In such cases, data transfers between spatially and temporally misaligned components are required so that they do not violate basic principles of hydrologic modeling (e.g., conservation of mass, energy, momentum). The goal of this research is therefore to study model coupling for cases where there are feedback loops between model components and those model components may run of different time steps requiring temporal interpolation of data exchanges.

To address this research goal, we conducted a hypothetical modeling exercise with the objective of estimating pollution concentration in a multimedia water/sediment system. In the first part of the study we simulated this system using two approaches: the first was a tightly-coupled numerical model and the second was as two loosely-coupled model components with one representing the water medium and the other representing the sediment medium. We used the first modeling approach to provide a point of comparison for the component-based modeling approach. The two components were coupled so that they exchanged concentrations across the water/sediment boundary through a bi-directional link during simulation runtime. In the second part of the study, we made the two components run on different time steps in order to study how interpolation schemes can be used to rescale data transfers between components during simulation runtime. Details of the experiments are provided in the following methods section of this paper. Following the methods section, we present results and discussion of the experiments including details of how the bidirectional link and data transformations work in OpenMI. Finally, we present conclusions from our work drawn from the investigation into these two aspects of using component-based modeling to simulate hydrologic systems.

2. Methods

2.1. Model Development

To better understand the application of component-based modeling for simulating fully coupled systems we considered a hypothetical case where water is above a sediment column, a constant source of a pollutant is injected into the water, and the pollutant is transported through the system by advection and diffusion. A two dimensional representation of the system can be described by the advection-diffusion equation

$$\frac{\partial C}{\partial t} = D_x \frac{\partial^2 C}{\partial x^2} + D_z \frac{\partial^2 C}{\partial z^2} - u \frac{\partial C}{\partial x} - v \frac{\partial C}{\partial z} \tag{1}$$

where C is the concentration of the pollutant (ppm) at a location (x, z) within the system at a time t, u is the velocity in x-direction, v is the velocity in z-direction (cm s^{-1}), and D_x and D_z are diffusion coefficients in the x and z directions (cm^2 s^{-1}), respectively. For simplicity, in the water medium we assumed only advective transport in the x-direction and diffusive transport in the z-direction, and in the sediment medium we assumed only diffusive transport in the z-direction. With these simplifications to Equation 1, the governing equation for the water medium becomes

$$\frac{\partial C_w}{\partial t} = D_{w,z} \frac{\partial^2 C_w}{\partial z^2} - u_{w,x} \frac{\partial C_w}{\partial x} \tag{2}$$

where C_w is the concentration of the pollutant in water (ppm), $D_{w,z}$ is the diffusion coefficient of the pollutant in water (cm^2 s^{-1}) in the z-direction, $u_{w,x}$ is the velocity of the pollutant in water (cm s^{-1}) in the x-direction. Similarly, the governing equation for the sediment medium after applying the simplifying assumptions becomes

$$\frac{\partial C_s}{\partial t} = D_{s,z} \frac{\partial^2 C_s}{\partial z^2} \tag{3}$$

where C_s is the concentration of the pollutant in sediment (ppm), $D_{s,z}$ is the diffusion coefficient of the pollutant in sediment ($cm^2\ s^{-1}$) in the z-direction. As described earlier, we solved the system using two approaches where in the first approach the water and sediment was modeled as a single system and in the second approach the water domain was modeled separately from the sediment domain but coupled through the shared boundary condition. Boundary conditions for the first approach for modeling the system were defined at the top of the water domain as $C_w(x, 0, t) = 1$ ppm for $0 \leq x \leq p$ where p is the length of the water domain (cm) that has a constant rate of pollutant injection (Figure 3). A transmissive boundary condition [*Riemann*, 2009] was used to describe the right, left, and bottom edges of the domain (details for the transmissive boundary condition are discussed after discretization of each domain governing equation). In the second approach for modeling the system, a boundary condition was introduced at the water/sediment interface that required $C_w(x, h, t) = C_s(x, h, t)$ to be satisfied, where h is the depth of the water column (cm) as shown in Figure 3. The initial condition of both approaches was $C(x, z, 0) = 0$.

Equations 2 and 3 were solved numerically for a two-dimensional grid in the x–z dimensions as a discrete representation of the water-sediment domain (Figure 3). We considered a simple domain consisting of ten rows (indexed by i's) and ten columns (indexed by j's) to conduct the case study experiment. The water medium was represented by the top row in the domain ($i = 0$), while the other nine rows represented the sediment medium. We approximated (Equation 2) using a finite difference approach with a forward explicit scheme

$$\frac{\Delta C_{0,j}}{\Delta t} = D_{w,z}\frac{C_{0,j} - 2C_{1,j} + C_{2,j}}{\Delta z^2} - u_{w,x}\frac{C_{0,j} - C_{0,j-1}}{\Delta x} \tag{4}$$

where $C_{i,j}$ is the pollutant concentration at location (i, j). Note that the w and s subscripts were not included for C in Equation 4 because the boundary condition will be controlled by the sediment domain. The transmissive boundary condition was used at the right edge of the water domain for both approaches and was approximated using a backward difference scheme such that $C_w(b, z, t) = C_w(b - \Delta x, z, t)$ where b is the domain width (cm) and Δx is the cell width (cm) in the computational grid (Figure 3). Concentration within the sediment domain was approximated using a finite difference leap frog (time marching) scheme as shown in Equation 5.

$$\frac{\Delta C_{i,j}}{\Delta t} = D_{s,z}\frac{C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} - 4C_{i,j}}{\Delta z^2} \quad (5)$$

Again the w and s subscripts were not included for C in Equation 5 because the boundary condition will be controlled by the water domain. This approach for approximating the governing equation was chosen because it has high stability for PDEs with oscillatory solutions [*Shampine*, 2009]. The transmissive boundary conditions used for the sediment domain for both approaches was approximated using a forward difference scheme for the left edge, $C_s(0, z, t) = C_s(\Delta x, z, t)$ for $h \leq z \leq d$ where d is the sediment domain height (cm), and a backward difference scheme for the right edge, $C_s(b, z, t) = C_s(b - \Delta x, z, t)$ for $h \leq z \leq d$. The sediment concentration at the bottom boundary was set to the concentration of the interior cell such that $C_s(x, d, t) = C_s(x, d - \Delta z, t)$ where Δz is the cell height (cm) in the computational grid. In solving the Equations 2 and 3, we assumed that the diffusion coefficients were constant over space and time. It is well known that a space explicit scheme does not oscillate when the Péclet Number ($P_e = v\Delta x/D$) is less than or equal to 2 and the Courant Number ($C_r = v\Delta t/\Delta x$) is less than or equal to 1.

Therefore we selected the parameter values used in the simulation (Table 1) to satisfy these conditions.

2.2. Model Implementation

We solved for concentration within the system using two approaches for structuring the code. In the first approach, which we will refer to as the *conventional approach*, the multimedia system was simulated using a tightly-integrated paradigm. By this we mean that the complete system, both the water and sediment medium, was represented within one code unit using a single multi-dimensional array. The numerical methods were programmed to operate on that array to solve Equations 4 and 5. The solution using this approach was implemented in Matlab. In the second approach, which we will refer to as the *component-based approach*, the multimedia system was simulated as two model components: one representing the water medium and the second representing the sediment medium. Each component was developed independently without knowledge of how the other component was implemented so that they remain autonomous units. The components were then linked together into a composition so that boundary conditions could be passed between the components during the simulation run. The solution using this approach was implemented in C# .Net using the version 1.4 of the OpenMI Software Development Kit (SDK) and the approach for simplifying the creation of new OpenMI-compliant models proposed by *Castronova and Goodall* [2010]. In the following paragraphs we provide further detail for the component-based modeling approach because it is the primary focus of this paper.

There were two steps in developing the component-based model: the first involved component development and the second involved linking components into a composition in

order to define how the components exchange data during a simulation run. Component-based software architectures rely on the standardization of component interfaces to provide the "plug-and-play" functionality where model components can be reused in multiple compositions without the need to change or recompile the model source code. The standard component interface, acts as a contract between the model component and the controlling application that runs the composition [*Szyperski*, 2002]. As stated earlier, we used the OpenMI as the component standard because it was designed specifically for water resource modeling. OpenMI provides the technical means for controlling a model component on a time-step basis. Our aim in this work was not to develop a component standard or composition orchestration application, but instead to use the existing tools provided through the OpenMI to model a hypothetical case study. We therefore built the components used in this study using the OpenMI standard in order to understand how this paradigm can be used to simulate a fully-coupled system.

To create the OpenMI components for this study, we used an approach for creating process-level OpenMI components described by *Castronova and Goodall* [2010] and named the Simple Model Wrapper (SMW). The approach automates many of the details involved in implementing the OpenMI standard so that the model developer can more easily and quickly create basic process-level hydrologic model components that adhere to the OpenMI standard. Using the SWM approach, the water and sediment components were defined by a configuration file, a geospatial dataset file, and a model engine file (Figure 4). The configuration file is an XML file that defines the metadata for each model component (see *Castronova and Goodall* [2010] for details). The input geospatial dataset file defines model elements, system parameters, initial state variables, and other attributes specific for each

model component. Finally, the model engine is a Dynamic Link Library (DLL) file that defines the behavior of the component and must implement three core methods: *Initialize*, *PerformTimeStep*, and *Finish*. The *Initialize* method reads the exchange items from the configuration file and the input parameters of each component from the geospatial dataset file and stores these properties in memory. The *PerformTimeStep* method advances the model component in time and performs the numerical calculations discussed earlier in the paper. Finally, the *Finish* method writes the output data for each component and releases the memory used by the components. The source codes for both components and tutorials describing how to create and use the components are available as part of the HydroModeler plug-in to the Consortium of Universities for the Advancement of Hydrologic Sciences, Inc. (CUAHSI) HydroDesktop software system (www.hydrodesktop.org).

The water and sediment components were linked into a composition by creating a bi-directional link between the two components. The bi-directional link specified that the sediment component required concentration values from the water component, and that the water component required concentration values for the elements in the top layer of the sediment component. Because OpenMI 1.4 follows a pull-driven communication paradigm, both the water and sediment components request values from the other component within the configuration on each time step. To start the simulation, we linked the sediment component to a trigger component which initiates component communication, as described earlier in this paper. When one component requests values from a linked component, that component is required to reply with the exact information requested. A request is made for values associated with a set of elements, a quantity, and a specific time step or time span. The component to which the request is made must supply values for the requested

elements and at the requested time step. If the requested data is not directly calculated by the component, as was the case in the second experiment we conducted where one component operated on a time step greater than the component that it was linked to, there must be an interpolation process to rescale data transfers.

For both the conventional modeling approach and the two experiments using the component-based modeling approach, which are described in the following subsections, we solved for $C(t)$ assuming a constant pollution point-source located in the first three nodes of water domain. The parameters for this study are given in Table 1 and the component configuration file attributes are given in Table 2. A constant velocity in the x-direction, u, and a constant rate of change with respect to time were given to avoid unsteady transport in the two-dimensional field. Interaction between the water medium and the sediment medium is through diffusive transport across the water/sediment boundary and therefore requires that the sediment component be able to obtain values calculated by the water component, and the water component be able to obtain boundary conditions from the sediment component during the model simulation run-time, as described earlier.

2.3. Experiment 1: Bidirectional Link

In the first experiment we used the component-based modeling approach to simulate the coupled system with a bi-directional link handling the feedback loop between the two components. The goal was to verify that the component-based model correctly simulated the system and to investigate how components coupled in a bi-directional link communicate with one another. The conventional modeling approach, where the system is solved for the entire domain with the same boundary and initial conditions, was used as a point of comparison to the component-based modeling approach. We compared the results of

the models at the first time step to ensure that the models were initialized with correct initial and boundary conditions, the tenth time step to check that the time marching computation and data exchange were occurring correctly, the 100^{th} time step to check the stability of the numerical computations in both components, and the 2000^{th} time step to judge the stability of the solution after a long time period. This comparison was done for time steps of 0.1 second, 0.5 second, 1 second and 10 seconds to ensure that both models were handling varying time steps correctly.

2.4. Experiment 2: Temporal Rescaling

In the second experiment we modified the component-based solution used in the first experiment so that the two components operated on different time steps. While components are required to have the same simulation time period in OpenMI, they are not required to have the same time step. To explore this feature of coupling temporally misaligned components, we varied the internal time step for the water component t_w from 1, 2, ..., 10 s, while holding the internal time step of sediment component t_s constant at one second. This change was implemented by changing the time step of the components as specified in the time horizon element of each component's XML configuration file (Table 3).

OpenMI compositions that have linked components operating on different space or time discretizations require interpolation in order to transform data exchanges between components. It is common for temporal interpolation algorithms to require values from one or more previous time steps in the simulation. To handle this case, the OpenMI includes a *SmartBuffer* class that is used to store values from previous time steps for a particular link in the computer's memory so that they can be accessed later when interpolated values are required (Figure 5). The functionality of a smart buffer is exposed

through three methods: the first is used to fill the buffer with values, the second to empty values from the buffer, and the third for obtaining interpolated values generated using a particular interpolation algorithm from the buffer.

The OpenMI Software Development Kit (SDK) provides a technique to interpolate between misaligned components that can be described by the following equation

$$S_{r,i} = \frac{S_{b,i}^{n+1} - S_{b,i}^{n}}{t_b^{n+1} - t_b^{n}}(t_b^r - t_b^n)(1-\alpha) + (S_{b,i}^n) \tag{6}$$

where t_b^n is the n^{th} entry in the buffered list, S_b^n is the n^{th} scalar set in the buffered list, α is the relaxation factor, and t_r is the requested time. If the relaxation factor (α) is zero, a linear interpolation is preformed. In the case where the relaxation factor is one, the return value is the nearest available value. For relaxation factors between one and zero, a weighted average between these two cases is returned [*Sinding et al.*, 2005].

We extended the OpenMI by introducing two other interpolation techniques and applied both new implementations to our study of the impact of interpolation on overall system mass balance. The first interpolation method that we implemented was the quadratic formula (Equation 7).

$$\begin{aligned} S_{r,i} = &\left[\frac{(t_b^r - t_b^n)(t_b^r - t_b^{n-1})}{(t_b^{n+1} - t_b^n)(t_b^{n+1} - t_b^{n-1})}\right](S_{b,i}^{n+1}) \; + \\ &\left[\frac{(t_b^r - t_b^{n+1})(t_b^r - t_b^{n-1})}{(t_b^n - t_b^{n+1})(t_b^n - t_b^{n-1})}\right](S_{b,i}^{n}) \; + \\ &\left[\frac{(t_b^r - t_b^{n+1})(t_b^r - t_b^{n})}{(t_b^{n-1} - t_b^n)(t_b^{n-1} - t_b^{n+1})}\right](S_{b,i}^{n-1}) \end{aligned} \tag{7}$$

The three $S_{b,i}$ values in Equation 7 can be applied for either uniform or nonuniform time steps. Because the water component was set to have a different time step compared to the sediment component, the storing capacity of the *SmartBuffer* was extended in order to perform this interpolation method.

The second interpolation method implemented was a cubic spline scheme whereby data exchanges are approximated by a curve in a piecewise manner by a third-order polynomial over each interval $t_b^n < t_b^r < t_b^{n+1}$. This is done in such a way that both the first and second derivatives of the curve at the end of the interval match those of the approximation of the immediate left at t_n and those to the approximation to the right at t_{n+1} . This can be expressed as

$$S_{r,i} = (t_b^r - t_b^n)\left[\frac{S_{b,i}^n + S_{b,i}^{n+1}}{(t_b^n - t_b^{n+1})} - (A+B)(t_b^{n+1} - t_b^n)\right] + C\frac{(t_b^r - t_b^n)^3}{6(t_b^{n+1} - t_b^n)} + D\frac{(t_b^r - t_b^{n+1})^3}{6(t_b^{n+1} - t_b^n)} \quad (8)$$

where A, B, C, and D are given as follows.

$$A = \frac{(S_{b,i}^{n-2} - 2S_{b,i}^{n-1} + S_{b,i}^n)}{(t_b^{n-2} - t_b^{n-1})^2}$$
$$B = \frac{(S_{b,i}^{n-1} - 2S_{b,i}^n + S_{b,i}^{n+1})}{(t_b^{n-1} - t_b^n)^2}$$
$$C = \frac{(S_{b,i}^{n-1} - 2S_{b,i}^n + S_{b,i}^{n+1})}{(t_b^{n-1} - t_b^n)^2}$$
$$D = \frac{(S_{b,i}^{n-2} - 2S_{b,i}^{n-1} + S_{b,i}^n)}{(t_b^{n-2} - t_b^{n-1})^2}$$

We used these two new interpolation methods implemented in OpenMI through this work (quadratic and cubic spline) along with the linear interpolation method provided through the OpenMI SDK to rescale data transfers between the sediment and water components in the composition. The goal of this second experiment was both to understand the data rescaling process and how it was handled by OpenMI, and also to quantify the impact of different interpolation routines on minimizing overall system mass balance in this specific case study for varying time step differences between the two model components.

3. RESULTS and DISCUSSION

The two experiments conducted through this research focus on two aspects of component-based modeling at its application to hydrologic systems. In the first experiment we explored how data is transferred between two model components linked with a bi-directional link using OpenMI. Bi-directional component communication represents a complicated case for component-based modeling, and we were interested in learning how this case is handled when modeling feedback loops. In the second experiment, we investigated the case where two components are linked but each component runs on a different time step. The OpenMI includes the concept of data transformations to couple spatially or temporally misaligned components. Our goal in this second experiment was to understand how different interpolation algorithms could be inserted into the OpenMI to rescale data and minimize overall system mass balance. For both experiments, we compared the component-based model to a second model of the same system that was implemented using a more conventional numerical scheme. The conventional model was used to ensure that the component-based models produced the same results and to quantify mass balance errors in the temporally misaligned model configuration.

3.1. Results from Experiment 1: Bidirectional Link

Results from the first experiment showed that the component-based model implementation and the conventional model implementation produced identical results for the simulation (Figure 6). The results were as expected. In the first time step, the concentration increased across the water medium due to advective transport. Then, as time progressed, the mass was transported by diffusion into the sediment due to the concentration gradient across the water/sediment interface. After ten seconds the concentration gradient

was parallel in the water medium and had reached a state where diffusion dominates the propagation of the pollutant in both media. The mass continued to be transferred into the sediment until the sediment reached steady-state so that the entire medium is saturated with the pollutant. The coupled component simulation was run for four different time steps (0.1, 0.5, 1, and 10 s) to investigate the sensitivity of the solution to time step size and also to ensure the numerical stability of the schemes used. When we used a time step that was smaller than one second, we had to change the *TimeEpsilon* attribute in the OpenMI Linkable Engine class of each component to be compatible with the corresponding time step of the model.

The protocol for communicating data between the two components was closely monitored in the experiment and is summarized in Figure 7. After the two components are initialized, the data exchange began with the trigger component requesting an exchange item from the first component in the chain: the water component in this case (Arrow 1 in Figure 7). The water component then requested concentration values for the boundary nodes of the sediment component for the initial time step in the simulation (Arrow 2 in Figure 7). The sediment component required boundary conditions from the water component, and therefore requested these values from the water component on the same time step (Arrow 3 in Figure 7). Time cannot advance until each component's data request has been answered, so at this point there was a deadlock where the sediment and water components were co-dependent on a shared boundary condition. Both of the components have not received their data requests, so they do not have sufficient information to compute the data requested and are therefore unable to advance in simulation time. This

is the challenge with a bi-directional link: each component requires data from the other component in order to step in time.

OpenMI handles deadlocks in bi-directional component linkages like the one described in the previous paragraph in the following way. First, the OpenMI standard requires that when a component has an unanswered request for a value, it is not allowed to issue any additional data requests. Second, OpenMI requires that a component always return values when a request for data is issued. Therefore in this experiment, when the water component requested data from the sediment component, the sediment component then requested data from the water component, and the water component must at that point answer that data request because it has an outstanding data request. The water component answers the data request issued by the sediment component with its best estimate for the concentration values at the water/sediment boundary at the current time step. In this case, the best estimate for values are the concentration values on the previous time step, as it is assumed that these values will be approximately equal to the concentration value on the current time step (Arrow 4 in Figure 7). At this point in the model run, the sediment component can answer the water component (Arrow 5 in Figure 7), and the water component is able to respond to the request for values issued by the configuration trigger (Arrow 6 in Figure 7). This concludes the first time step of the model configuration. The trigger then invokes the water component for the next time step, requesting data for that time step. The same interaction between the sediment and water component is repeated for this time step, and the model continues until all time steps have been completed.

This experiment shows that the OpenMI version 1.4 SDK handles deadlocks in bidirectional links by having the component estimate a value on the current time step based

on the values of that same variable calculated for the previous time step. We were able to reproduce the component-based model solution using a more conventional numerical solution by making this same assumption in the conventional numerical algorithm. For some cases, the assumption that current conditions can be approximated by past conditions may not be sufficient, in which case modification of the OpenMI SDK to produce a more sophisticated means for handling component bi-directional links may be necessary. For example, while we used an explicit scheme for the two model components in the experiment, an alternative approach would have been to implement the components using an implicit scheme. An implementation using an implicit scheme would have required iteration within a time step to obtain the correct values for the boundary conditions. The OpenMI Technical Association has created a utility package designed to support advanced data control of model compositions, such as iteration within a time step, that is available in a prototype form but not as part of the official OpenMI SDK. The utility package includes three data control options: iteration, calibration, and logic switch. The iteration controller is a linkable component that acts as a mediator between components and requests the models to step back one time step to adjust the values in the implicit scheme. The input exchange item of the receiving component and the output exchange item of the providing component should be connected to the same iteration controller component. Because we choose to use an explicit scheme when implementing the components, it was not necessary to use this prototype utility package for advanced control of OpenMI compositions.

3.2. Results from Experiment 2: Temporal Rescaling

In the second experiment performed as a part of this work, we studied how temporal misalignments of model components, and more specifically interpolation of component data transfers, are handled by OpenMI. Introducing different interpolation techniques to the OpenMI and using existing interpolation techniques provided within the OpenMI SDK, we were able to quantify the impact of interpolation on system mass balance error. To explore this topic, we relaxed the assumption used in the first experiment that the water and sediment components operate on the same time step. This introduces the need to rescale data exchanges between the two components using interpolation schemes. In this second experiment we were interested in better understanding how the case of temporal misaligned of component data exchanges is handled within the OpenMI paradigm. We were also interested in learning how to include new interpolation algorithms into the OpenMI system, and how these different interpolation methods influence the overall mass balance error within the coupled system of components.

Figure 8 depicts the protocol OpenMI uses to rescale data transfers by demonstrating the steps involved in one cycle of sediment request for interpolated data from water component. At the point in time which the figure depicts, both components have been initialized and the water component has completed its first time step of the simulation t_w^1. First, the sediment component attempts to compute values for the current time step of the simulation run ($t_s = t_s + \Delta t_s$). In order to do this calculation, the sediment component requires values from the water component and so it issues a data request to the water component for t_s. The water component is then required to return values of the water concentration for time equal to t_s^1. The water component will evaluate if the requested

time is before, after, or equal to its own internal time. The water component determines in this case that the requested time is less than its internal time ($t_s < t_w$), therefore the water component will request the *SmartBuffer* to interpolate values at time step t_s. Finally the *SmartBuffer* returns the data values back to the sediment component. Thus, the *SmartBuffer* object is central in the interpolation process required to couple spatially or temporally misaligned model components.

As stated earlier, the OpenMI SDK provides an interpolation algorithm that can be used to implement linear interpolation, nearest neighbor interpolation, or a weighted combination of these two interpolation algorithms. For some cases, these options may not be optimal for rescaling data between spatially or temporally misaligned model components. Many hydrological flux and state variables follow a polynomial behavior, for example, and therefore it would be necessary to have such interpolation methods available as a part of the OpenMI toolkit. OpenMI is developed as an open source project, so we were able to extend the set of interpolation schemes available through OpenMI to include both the quadratic polynomial and cubic spline schemes as described in the Methods section.

After adding these new interpolation algorithms to OpenMI, we applied them to handle the case where the water and sediment components had different time steps. In linear interpolation, the *SmartBuffer* was required to store only two values to perform the interpolation (Figure 8). Therefore, as time advances, the *SmartBuffer* was updated with the new value as it became available. This means that the *SmartBuffer* was acting as a moving frame only storing two values at any given time. More complicated interpolation methods require the *SmartBuffer* to store more than two data values in the buffer at any given time. This could have performance costs in terms of increased search time on the

smart buffer and increased storage required for the buffer, however this aspect of the work was not investigated in detail as part of this study. We did not notice any measurable performance costs in this work, however this was a purposely small modeling case study and so we cannot rule out the possibility of performance costs for a larger modeling exercise.

To understand the benefit of each interpolation scheme in minimizing the mass balance error between the temporally misaligned coupled components, the internal time step of the sediment component (Δt_s) was fixed at one second and the water component internal time step (Δt_w) was varied from two through ten seconds with a step of one second. Three interpolation schemes – linear, quadratic, and natural cubic spline – were used to interpolate between the previous stored values and the current value of the water component value at an intermediate time equal to t_s before the values are passed back to the sediment component so that it can proceed in its own calculations. Figure 9 represents the impact of each interpolation scheme on the computed sediment boundary concentration. The comparison was done using the concentration values of the sediment boundary layer calculated when both the water and sediment components were operating on the same internal time step and the concentration values calculated using interpolation schemes separately when both components were temporally misaligned. The outputs obtained from the temporally aligned component configuration are shown on the horizontal axis and the predicted (interpolated) outputs from the misaligned component configuration are shown on the vertical axis. The results are presented at three different locations in the sediment top layer (x = 4, 6, and 10 cm), and for three different Δt_w values (2, 4, and 10 s). The figure shows that the interpolation error increases as the distance from the source pollution source increases so that the error recorded at point ($x = 10$ cm) is greater than

the error recoded at point ($x = 2$ cm). The cubic spline scheme best matches the actual values for small differences in time steps between the water and sediment components for all three locations. As the difference between the operating times steps of the components increase, the deviation of the interpolated values using the three interpolation schemes also increases. The linear scheme shows the best results for large differences in temporal time step sizes between components.

Figure 10 shows the total mass error in the sediment media as the time advances during the simulation run. Total mass error represents the difference between total mass of the sediment media when both components are temporally aligned and the predicted mass when the components are misaligned with the three differences in time steps (2, 4, and 10 s). The results show that the cubic scheme minimizes the error in mass transfer between coupled components the best for the first two cases ($\Delta t_w = 2$ s and $\Delta t_w = 4s$), but does the worst for the third case ($\Delta t_w = 10s$). For this third case, a linear interpolation is best at minimizing mass balance error in the misaligned component configuration. In general we found that for smaller time steps, linear interpolation has the largest error while cubic spline interpolation results in the smallest error. We also found that the opposite is true for larger time steps with spline interpolation producing the largest error and linear interpolation producing the smallest error. The reason for this finding is that an interpolated concentration value is estimated from the previous time step value of the node, and the previous time step value of the spatially adjacent nodes. For the case of the linear interpolation scheme, only the previous value is used in the interpolation, while the quadratic scheme uses the two previous values. The cubic spline interpolation scheme is more complicated in that it uses a piece-wise interpolation that employs a third order

polynomial between each of the subintervals $[t_i, .., t_{i-3}]$ in order to satisfy continuity at junction points between curve segments and continuity in its first and second derivatives. This high dependence on neighboring values in space and time explains the difference in the interpolated values calculated using cubic spline scheme when compared to the values calculated using linear scheme for large time steps. Furthermore, for the first few time steps, the cubic spline interpolation is actually performing a linear interpolation because there are not a sufficient number of previous values to perform a cubic spline interpolation. Then, once a sufficient number of previous values are available to perform the cubic spline interpolation, the interpolation scheme is able to minimize mass balance error between the two coupled but temporally misaligned model components. This fact explains the sudden decrease in the mass balance error after the third time step when using the cubic spline interpolation for ($\Delta t_w = 2$ s and $\Delta t_w = 4$ s).

4. SUMMARY and CONCLUSION

While previous work has demonstrated the benefits of using a component-based modeling design for simulating hydrologic systems, there remain important research questions about the applicability of the approach for modeling complicated system dynamics. This work provides a detailed view of two aspects of component-based modeling relevant for simulating hydrologic systems: feedback loops and misalignment of data exchanges. We explored these aspects specifically for the OpenMI component-based modeling protocol because it was designed and development for the water resource modeling community. The topics were explored through a simplified case study of mass transport within a mixed media system with water over a sediment column. A component-based implementation of this system was compared with a more conventional numerical solution to the same

system. This comparison provided a means for understanding how the component-based composition was solved, and for quantifying mass balance errors in the case where component output needs to be temporally rescaled to accommodate the input needs of another component.

In the first experiment conducted where we examined how feedback loops (or what OpenMI terms bi-directional links) between components are handled, we found that OpenMI handles such cases in the following way. First, when a component has an unanswered data request, it is not allowed to issue any additional data requests. Second, a component is required to always return values when a request for data is issued. These two requirements result in components having to estimate values based on previous values calculated by the component when two components are coupled with a bidirectional link. The design of OpenMI makes it possible to enhance the logic for handling feedback loops by adopting a scheme that allows for iteration between components to converge on a shared boundary condition. One finding of this research is that such a solution may be required for cases where model components have large time steps, for example, where it is insufficient for the component to reply to a request for values on the current time step with the values from the previously time step as its best estimate. Future research is needed to more fully explore possibilities for more sophisticated solutions like those available in the prototype advanced controller utilities package to handle iteration between components coupled through a bi-directional feedback loop.

In the second experiment where we examined how misaligned component interactions are handled within OpenMI, we found that the framework provided an open architecture whereby new interpolation algorithms could be easily added in order to diversify the

schemes available to users for rescaling transfers between components. For the particular advection-diffusion case study, we found that a cubic spline interpolation was best suited for minimizing system mass balance error for cases where there is a small time step difference between model components, whereas a linear interpolation preformed best for large time step differences between model components. An important point is that multiple rescaling techniques can be made available through the OpenMI standard and so the user can select the most appropriate approach for a specific application, or add a new approach to the system if necessary. An alternative approach if the users cannot define the appropriate scheme for their application is testing the performance of each scheme following the methodology in our second experiment. Open source software and open modeling architectures also allow modelers to share not only model components, but also interpolation routines and other useful tools that are necessary to modeling hydrologic systems. While this is possible in OpenMI version 1.4, OpenMI version 2.0 provides a more flexible approach for adapting output data to fit the input requirements of another model by introducing the AdaptedOutput construct for rescaling data in space or time, or performing on-the-fly unit conversions between linked models. Users could create a time interpolation adapted output construct and easily share it with others for reuse in component-based modeling applications.

In conclusion, component-based modeling, were a complex systems is decomposed into a set of simpler components that act as separate but linked units, presents many benefits for modeling hydrologic systems. Each component can be designed, developed, and maintained by different groups, but still be used by a large community of modelers in their own simulations. OpenMI provides an implementation of component-based model-

ing specifically for the water resources community. This work provides a detailed look at how component-based modeling in general and OpenMI in particular handle what we consider to be two of the more complicated challenges in representing hydrologic systems as coupled components. Feedback loops are handled in a fairly simplistic way in the current implementation of the OpenMI SDK, although more complicated schemes that allow for iteration between components that are coupled through a feedback loop dependency. The OpenMI includes a sophisticated means for handling data transfers between components that do not have the same spatial or temporal discreteization. While the framework only includes a few interpolation methods, it is possible to add new interpolation algorithms, as we have done in this work, that plug into the system. Nonetheless, users of component-based architectures must understand potential errors introduced when coupling spatially or temporally misaligned models. This work is an attempt to quantify such errors for a specific case study and to understand how different interpolation routines can be used to minimize errors.

Acknowledgments. The authors wish to acknowledge the National Science Foundation (NSF) for supporting this research under the award CBET: 08-4244 "CAREER: Integrated Modeling for Watershed Management."

References

Armstrong, R., D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski, Toward a common component architecture for high-performance scientific computing, in *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, pp. 115–124, IEEE, 2002.

Beven, K., Towards an alternative blueprint for a physically based digitally simulated hydrologic response modelling system, *Hydrological processes*, *16*(2), 189–206, 2002.

Bloschl, G., and M. Sivapalan, Scale issues in hydrological modelling: A review, *Hydrological Processes*, *9*(3-4), 251–290, doi:10.1002/hyp.3360090305, 1995.

Bulatewicz, T., X. Yang, J. Peterson, S. Staggenborg, S. Welch, and D. Steward, Accessible integration of agriculture, groundwater, and economic models using the Open Modeling Interface(OpenMI): methodology and initial results, *Hydrology and Earth System Sciences*, *14*(3), 521–534, 2010.

Castronova, A. M., and J. L. Goodall, A generic approach for developing process-level hydrologic modeling components, *Environmental Modelling & Software*, *25*(7), 819–825, doi:10.1016/j.envsoft.2010.01.003, 2010.

Clements, P. C., From subroutines to subsystems: Component-based software development, in *Proceedings of the 17th international conference on Software engineering - ICSE '95*, pp. 179–185, Pittsburgh, Pennsylvania, USA, doi:10.1145/225014.225031, 1995.

Dozier, J., Opportunities to improve hydrologic data, *Reviews of Geophysics*, *30*(4), 315–331, 1992.

Ewert, F., et al., A methodology for enhanced flexibility of integrated assessment in agriculture, *Environmental Science & Policy*, *12*(5), 546–561, 2009.

Fotopoulos, F., C. Makropoulos, and M. Mimikou, Flood forecasting in transboundary catchments using the Open Modeling Interface, *Environmental Modelling & Software*, *25*(12), 1640–1649, doi:10.1016/j.envsoft.2010.06.013, 2010.

Garlan, D., R. Allen, and J. Ockerbloom, Architectural mismatch or why it's hard to build systems out of existing parts, in *Proceedings of the 17th international conference on Software engineering - ICSE '95*, pp. 179–185, Seattle, Washington, United States, doi:10.1145/225014.225031, 1995.

Goodall, J., and D. Maidment, A spatiotemporal data model for river basin-scale hydrologic systems, *International Journal of Geographical Information Science*, *23*(2), 233–247, 2009.

Goodall, J., J. Horsburgh, T. Whiteaker, D. Maidment, and I. Zaslavsky, A first approach to web services for the National Water Information System, *Environmental Modelling & Software*, *23*(4), 404–411, 2008.

Gössler, G., and J. Sifakis, Composition for component-based modeling, in *Formal Methods for Components and Objects*, pp. 443–466, Springer, 2003.

Gregersen, J. B., P. J. A. Gijsbers, and S. J. P. Westen, OpenMI: Open Modelling Interface, *Journal of Hydroinformatics*, *9*(3), 175, doi:10.2166/hydro.2007.023, 2007.

Heineman, G., and W. Councill, *Component-based software engineering: putting the pieces together*, vol. 17, Addison-Wesley USA, 2001.

Horsburgh, J., D. Tarboton, D. Maidment, and I. Zaslavsky, A relational model for environmental and water resources data, *Water Resources Research*, *44*(5), W05,406, 2008.

Leavesley, G. H., S. L. Markstrom, M. S. Brewer, and R. J. Viger, The modular modeling system (MMS) -The physical process modeling component of a database-centered decision support system for water and power management, *Water, Air, & Soil Pollution*, *90*(1-2), 303–311, doi:10.1007/BF00619290, 1996.

Maidment, D., Bringing water data together, *Journal of Water Resources Planning and Management*, *134*, 95, 2008.

Miller, R. C., D. P. Guertin, and P. Heilman, Information technology in watershed management decision making, *Journal of the American Water Resources Association*, *40*(2), 347–357, doi:10.1111/j.1752-1688.2004.tb01034.x, 2004.

Pullar, D., and D. Springer, Towards integrating GIS and catchment models, *Environmental Modelling and Software*, *15*(5), 451–459, 2000.

Rahman, J. a., It's time for a new environmental modelling framework., *Journal of the American Water Resources Association*, *4*(5723), 1727–1732, doi:10.1126/science.1110411, 2003.

Reussner, F., J. Alex, M. Bach, M. Schütze, and D. Muschalla, Basin-wide integrated modelling via OpenMI considering multiple urban catchments, *Water science and technology*, *60*(5), 1241–1248, 2009.

Shampine, L., Stability of the leapfrog/midpoint method, *Applied Mathematics and Computation*, *208*(1), 293–298, doi:10.1016/j.amc.2008.11.029, 2009.

Sinding, J. B. Gregersen, P. J. A. Gijsbers, R. Brinkman, and S. J. P. Westen, the OpenMI Document Series: Part F org.openmi.utilities technical documentation (version 1.0).

Singh, V., *Watershed models*, Taylor & Francis, Boca Raton, 2006.

Srinivasan, R., and J. G. Arnold, Integration of a Basin-Scale Water Quality Model With GIS, *Journal of the Astronomical Society of Western Australia*, *30*, 453–462, 1994.

Steward, D. R., J. M. Peterson, X. Yang, T. Bulatewicz, M. Herrera-Rodriguez, D. Mao, and N. Hendricks, Groundwater economics: An object-oriented foundation for integrated studies of irrigated agricultural systems, *Water Resources Research*, *45*(5), doi:

10.1029/2008WR007149, 2009.

Syvitski, J. P., E. W. Hutton, and S. D. Peckham, CSDMS – A Modeling System to Aid Sedimentary Research and Rudy Slingerland, *The Sedimentary Record*, *9*(1), 4–9, doi:doi: 10.2110/sedred.2011.1.4, 2011.

Szyperski, C., *Component software : beyond object-oriented programming*, 2nd ed. ed., ACM Press, Addison-Wesley, New York ,London ,Boston, 2002.

Toro, E., 2009, Riemann solvers and numerical methods for fluid dynamics, 3rd *Edition*, *Springer*, New York, doi:10.1007/9783540498346.

Tindall, C., An overview of the open modelling interface and environment (the OpenMI), *Environmental Science & Policy*, *8*(3), 279–286, doi:10.1016/j.envsci.2005.03.009, 2005.

Voinov, A., C. DeLuca, R. Hood, S. Peckham, and C. Sherwood, A Community Approach to Earth Systems Modeling, *EOS*, *91*(13), doi:10.1029/2010EO130001, 2010.

Voinov, A. and C. Cerco, Model integration and the role of data,*Environmental Modelling & Software*, *25*(8), 965–969, doi:10.1016/j.envsoft.2010.02.005, 2010.

Wurbs, R., Dissemination of generalized water resources models in the united states, *Water International*, *23*(3), 190–198, doi:10.1080/02508069808686767, 1998.

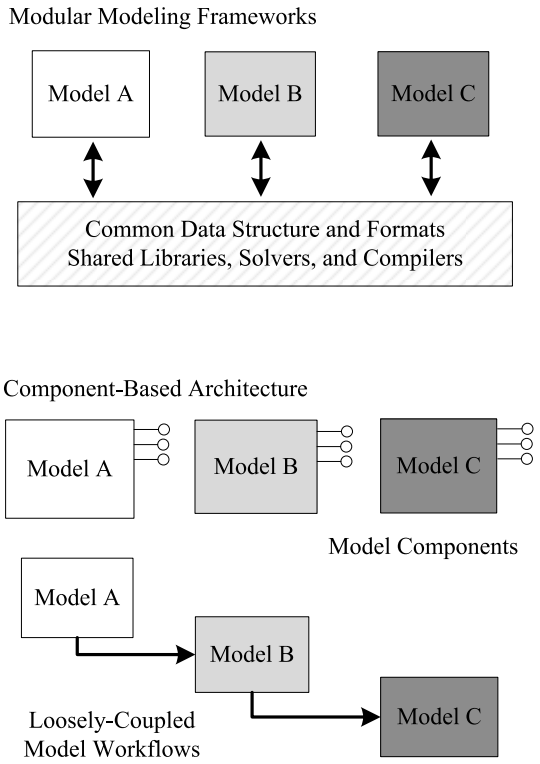

Figure 1. Contrasting the more common modular approach for structuring model codes with a component-based approach.

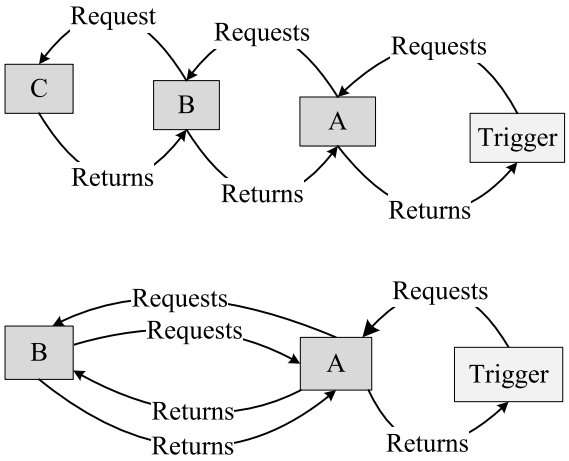

Figure 2. Data exchange between component models in a uni-directional composition (top) and a bi-directional composition (bottom) (adapted from [*Tindall*, 2005])

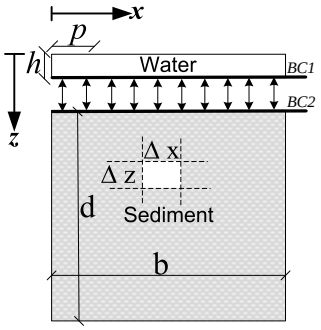

Figure 3. The water and sediment domains used in the component-based approach. $BC1$ is the water component lower boundary values that act as the sediment component top boundary, $BC2$ is the sediment component top boundary values that acts as the water component lower boundary.

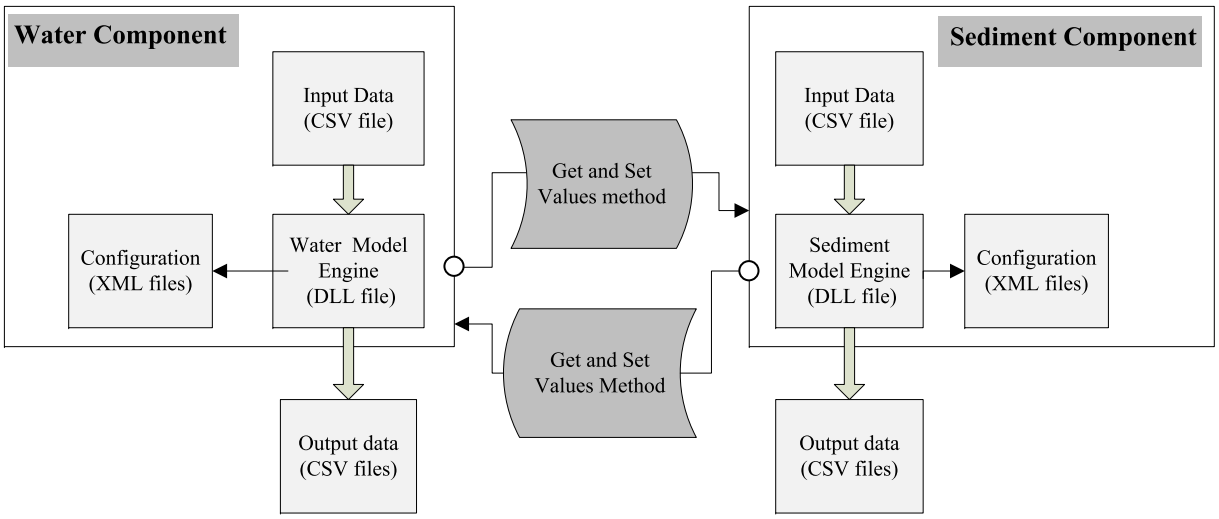

Figure 4. Details for the implementation of the component-based model approach.

	Water	Sediment
Velocity (cm s^{-1})	0.80	0.00
Diffusion Factor (cm^2 s^{-1})	2.5×10^{-4}	2.5×10^{-4}
No. of Rows	1	9
No. of Columns	10	10

Table 1. Model parameters for water and sediment media

Component Name	Exchange Item Type	Element Set Name	Quantity Name	Units
Water Component	Output Item	ConcentrationBC1	Water Concentration	ppm
	Input Item	ConcentrationBC2	Sediment Concentration	ppm
Sediment Component	Input Item	ConcentrationBC1	Water Concentration	ppm
	Output Item	ConcentrationBC2	Sediment Concentration	ppm

Table 2. Properties included in the XML configuration files for the water and sediment components.

Start Date Time:	08/20/2009 00:00:00 AM
End Date Time:	08/20/2009 01:00:00 AM
Time Steps:	0.1, 0.5, 1 and 10 s

Table 3. The time horizon and time steps used in the simulation.

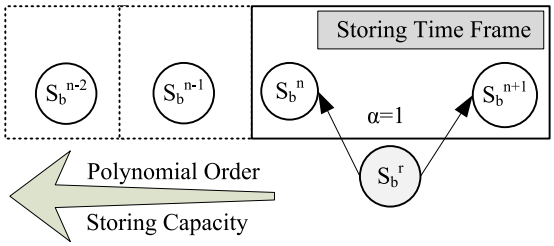

Figure 5. Provided interpolation routine in OpenMI which allows for linear interpolation (when $\alpha = 0$), nearest neighbor (when $\alpha = 1$), and a weighted approach between these two alternatives (when $0 < \alpha < 1$)

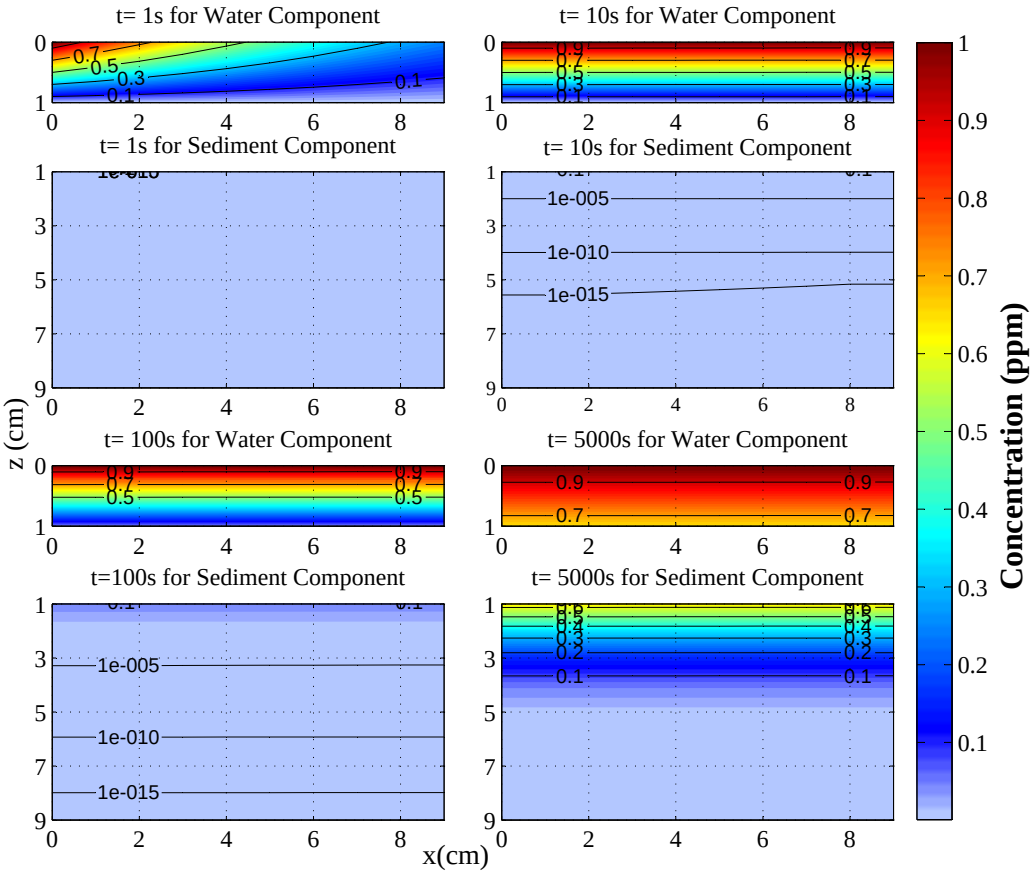

Figure 6. Results of the coupled water/sediment system for both the conventional and component-based approach.

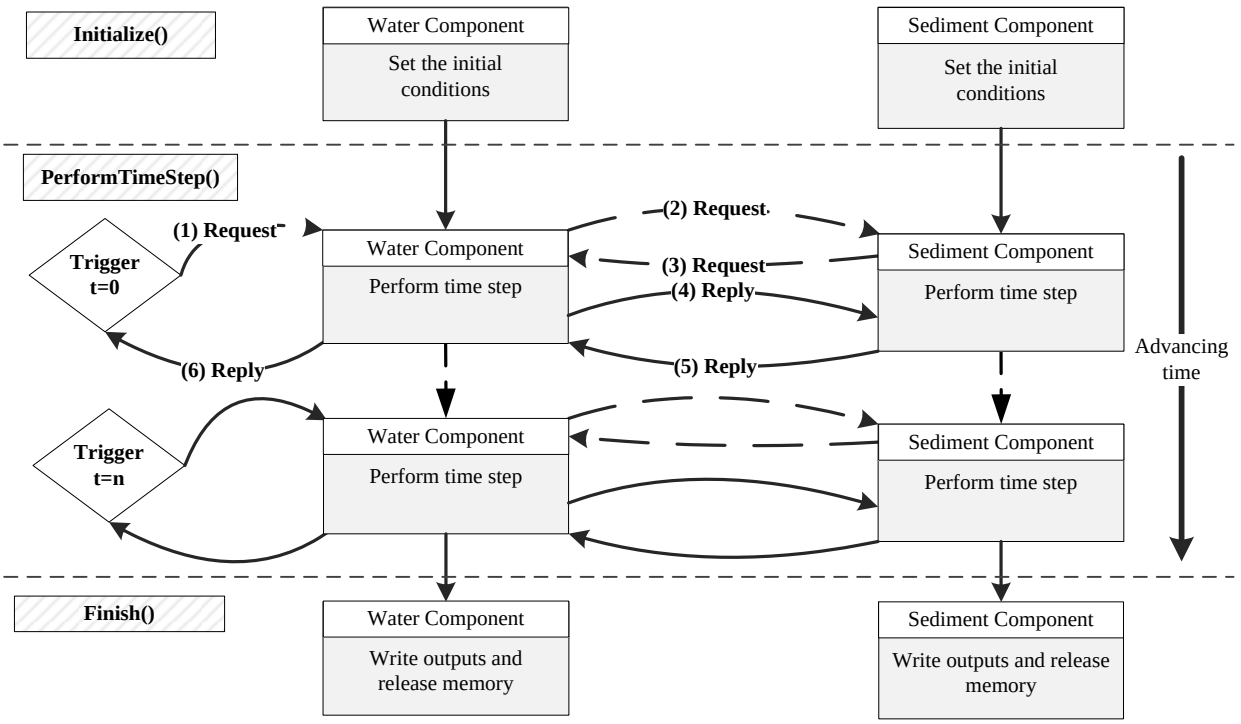

Figure 7. Communication flow for a bidirectional link within OpenMI between the water and sediment components (adapted from *Sinding et al.* [2005]).

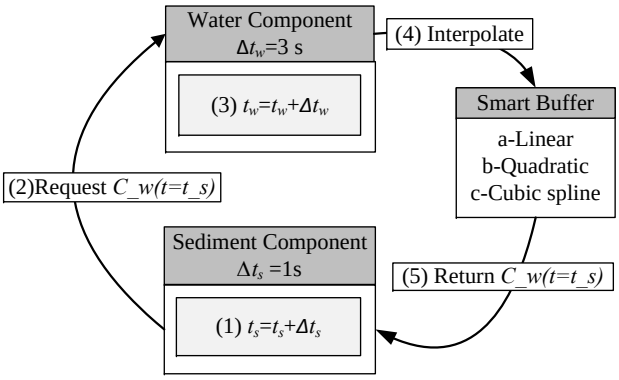

Figure 8. Communication flow for rescaling data transfers when components inputs and outputs are temporally misaligned.

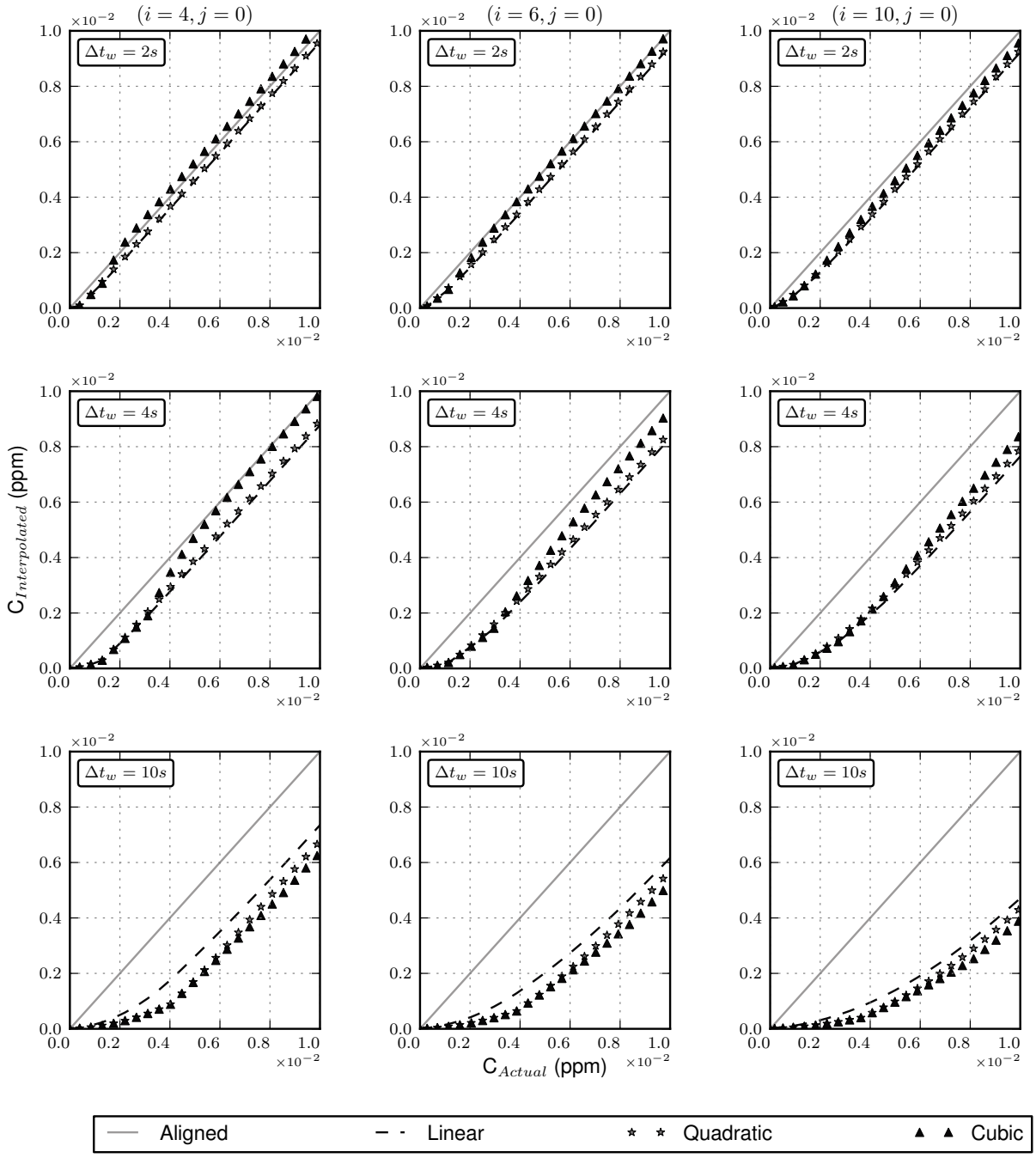

Figure 9. Results of linear, quadratic, and cubic spline schemes in minimizing the mass balance error between temporally misaligned coupled components. Comparison is for three locations in the sediment top layer (x = 4, 6, and 10 cm) and for three values of internal time step differences (2, 4, and 10 s).

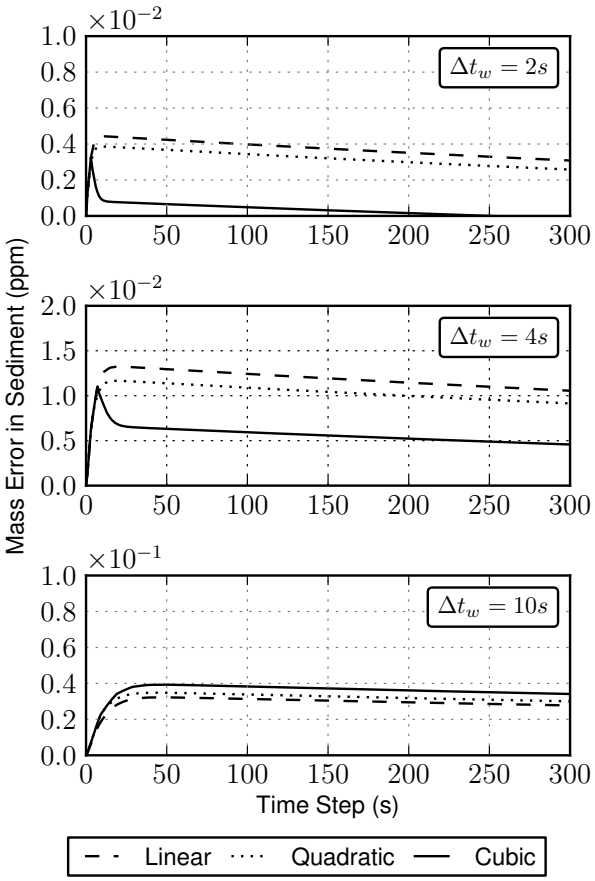

Figure 10. Total mass error in sediment domain between the aligned component configuration and the misaligned component configurations coupled using a linear, quadratic, and cubic spline scheme for three values of internal time step differences (2, 4, and 10 s).