This is an Accepted Manuscript of an article published in Environmental Modelling and Software in 2013 available online: http://dx.doi.org/10.1016/j.envsoft .2012.02.011

Integrated Modeling within a Hydrologic Information System: An OpenMI Based Approach

Anthony M. Castronova[a], Jonathan L. Goodall[a], Mehmet B. Ercan[a]

[a]*Department of Civil and Environmental Engineering*
University of South Carolina
300 Main Street, Columbia, South Carolina 29208 USA

Abstract

This paper presents a prototype software system for integrated environmental modeling that provides interoperability between the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) and the Open Modeling Interface (OpenMI). The primary motivation for making these two systems interoperable is that the CUAHSI HIS has a primary focus on hydrologic data management and visualization while the OpenMI has a primary focus on integrated environmental modeling. By combining the two systems into a single software application, it is possible to create an integrated environmental modeling environment that scientists and engineers can use to understand and manage environmental systems. Using standards to achieve the steps required to find, gather, integrate, and analyze hydrologic data allows for a wide community of groups to participate because it establishes key rules and protocols that must be followed in order to add to the overarching system. The key contribution of this work, therefore, is an investigation of two standards in the community and exploring ways to provide interoperability between them. HydroModeler is a software implementation of our work and provides an OpenMI-compliant modeling environment embedded within the CUAHSI HIS HydroDesktop software system. We describe the design and implementation of this prototype software system, and then present an example application in which evapotranspiration is modeled using OpenMI components that consume HIS time series data for input. Finally, we conclude with a summary of our experience exploring the potential for interoperability between data and modeling systems, and suggest ways in which future development can

better facilitate connections between the various subsystems needed within an integrated environmental modeling system.

Keywords: Integrated Modeling, Data Management, Systems Analysis, Environmental Management

1. Introduction

Environmental management often requires both observations and models to answer policy questions and to address potential or current problems. It is therefore important to consider approaches for using data management systems in combination with models to study environmental systems. While there are many examples of data management and modeling systems as separate tools (Syvitski et al, 2004; Moore and Tindall, 2005; Kralisch et al, 2005), there are fewer examples of integrated systems capable of handling both of these activities (Argent et al, 2009). Furthermore, the general trend toward standardization in both the data and modeling communities suggests a path forward for combining existing tools that are built from established data transmission and communication standards. This integration would allow for a broad community of individuals and groups to contribute to an environmental management system.

This paper focuses on two existing technologies, the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) and the Open Modeling Interface (OpenMI), and explores how they can be combined to create a complete environmental management system. The CUAHSI HIS has been developed with the goal of enhancing access to hydrologic data (Maidment, 2008; Tarboton et al, 2009). Concurrent to this effort, the Open Modeling Interface (OpenMI) Association has developed a standard to facilitate model coupling and a reference Software Development Kit (SDK) for implementing the standard (Moore and Tindall, 2005; Gregersen et al., 2007). Because the two systems were developed by independent groups, there is no formal mechanism for using both the HIS and the OpenMI together. However, the systems share important similarities that make interoperability possible, as demonstrated in this paper.

The objective of this research is to explore how interoperability between the CUAHSI HIS and the OpenMI can be achieved, and then to use this knowledge to design and prototype a software application that demonstrates system interoperability. The prototype software application, named HydroModeler, is an integrated environmental modeling environment implemented as a plug-in to the CUAHSI HydroDesktop software system (Ames et al., 2009) in order to allow for OpenMI-compliant modeling within the HIS. HydroModeler supports any OpenMI-compliant (Microsoft .NET Framework 4.0) model and enables users to create model configurations where data is supplied by the HIS into simulations and, likewise, data can be written back from a simulation into a local data repository. This data interoperability is possible using two new OpenMI components, a database reader and a database writer. Furthermore, this functionality enables other HydroDesktop tools to work with model output. For example, the HydroDesktop charting and mapping views provide temporal and spatial visualization capabilities for model outputs.

In the following section we provide further background on the CUAHSI HIS and OpenMI to familiarize the reader with these two technologies. We then present our approach for integrating the HIS and OpenMI, including a summary of the challenges encountered and a discussion of alternative approaches considered. We next present HydroModeler as a prototype application that provides the ability to build and execute OpenMI model configurations that leverage HIS data. An example study is then used to showcase how these systems can be applied to model a hydrologic process. This example study demonstrates a small piece of what could be a much larger environmental or cross-disciplinary model. Finally, we conclude with a summary of the research results and a brief discussion of future research plans.

2. Background

2.1. CUAHSI Hydrologic Information System (HIS)

The HIS can be viewed as three separate but interconnected subsystems: HydroServer, HIS Central, and HydroDesktop (Figure 1)(Tarboton et al, 2009). HydroServer is a data sharing tool provided as part of the CUAHSI HIS software stack (Horsburgh et al., 2009). It includes a database schema, known as the Observations Data Model (ODM), for storing observational time series (Horsburgh et al., 2008). In a HydroServer, an ODM database is exposed using the WaterOneFlow web service Application Programming Interface (API), and software tools are provided for managing time series data within an ODM database (Tarboton et al, 2009). HIS Central is a metadata catalog that enables search across distributed HIS data. It includes an ontology and controlled vocabulary to mediate semantic heterogeneity across multiple data providers. In basic terms, the ontology provides the structure needed to integrate disparate systems (i.e. data from different sources) and the controlled vocabulary establishes the precise language needed for inter-system communication (Gruber, 2009). Lastly, the HydroDesktop is a desktop application that enables end users to search, download, and analyze hydrologic data available through HIS Central (Ames et al., 2009). It utilizes a back-end database with a schema similar to the ODM for storing observation data on the user's local machine. The HydroDesktop Graphical User Interface (GUI) is built on an open source Geographic Information System (GIS) platform named MapWindow GIS (Ames et al., 2008) that allows for the extension of core functionality through plug-in software. Plug-in extensions can be developed in the C# (Microsoft .NET Framework 4.0) programming language using a HydroDesktop plug-in interface standard. HydroModeler is one such plug-in extension that adds integrated modeling capabilities to HydroDesktop.

The CUAHSI HIS follows a service oriented architecture (Curbera et al., 2002; Huhns and Singh, 2005) because each of the three systems described in Figure 1 are interconnected by web services (Tarboton et al, 2009). Hydrologic data is stored in databases throughout the world and are exposed on the Internet using web service standards (Goodall et al., 2008; Tarboton et al, 2009). The HydroDesktop application, for example, obtains metadata from the HIS Central system using web services to identify available datasets. A second set of web services, called WaterOneFlow, are used to obtain these datasets from specific instances of HydroServers, or any other database that is exposed using the HIS web service standards (Horsburgh et al., 2009). This design principle allows the overall HIS architecture to be open and extensible. For example, third party applications that require access to hydrologic data can communicate directly with HIS Central or HydroServer systems, using their respective web services. Moreover, a model can obtain input data directly from a HydroServer, rather than using the graphical HydroDesktop application to prepare input files (e.g. Billah and Goodall, 2011).

2.2. Open Modeling Interface (OpenMI)

The OpenMI is a standard that defines how models exchange data during a simulation run (Moore and Tindall, 2005). It is accompanied by a reference Software Development Kit (SDK) that provides tools for implementing the standard to perform integrated environmental modeling (Gregersen et al., 2007). This research uses OpenMI version 1.4, the current release during the time that the majority of the research was conducted. The OpenMI standard consists of interfaces that can be used to couple models so that they are able to seamlessly exchange data during run time. For example, an integrated modeling effort may require coupling watershed, river hydraulics, and groundwater models, as shown in Figure 2. The OpenMI enables such models to be coupled and exchange data necessary to simulate system interactions and dependencies. This approach enables each model to maintain its own identity so that the model can also run independently as well as within a larger system. Therefore, the OpenMI can be described as a loose integration software architecture (Gregersen et al., 2007) and is in contrast to tight integration approaches where the models are combined into a single system (e.g. Yu et al, 2006; Maxwell et al., 2007; Ahrends et al, 2008). Loose integration implies that models are coupled in a "plug-and-play" manner, such that it is possible to reconfigure how they interact without recompiling the source code (Argent, 2004). While the OpenMI was designed to couple large legacy models for environmental management, it is also possible to create configurations from new components created for research purposes (Bulatewicz et al., 2009; Castronova and Goodall, 2010). One of the most attractive features of component-based modeling is that specific parts of a model system can be interchanged to test their individual impact. This aspect in particular makes the approach useful for scientific research and instruction.

The OpenMI concept of a linkable component can be used to couple models, databases, web services, file directories, or any other resource that needs to share data with external components during a model simulation. Authoring and executing a component-based model is mediated by a configuration editor (Gregersen et al., 2007). The OpenMI Association offers a Standard Development Kit (SDK) that includes a basic configuration editor, called the OpenMI Configuration Editor (OmiED). This editor follows the "request-and-reply" communication paradigm defined by the OpenMI standard to achieve system integration (Gregersen et al., 2007). When using a component-based approach, the modeler defines a model configuration that specifies how components within the system are linked together. For example, a forcing variable such as precipitation might be stored within a database and made available to a rainfall/runoff model as a boundary condition, with OmiED orchestrating the data transfer between the database and model. The advantage of modeling a hydrologic system in this manner is that, once the coupling has been defined, each component can evolve separately from other components within the system as long as the standard interface specification is maintained (Argent, 2004).

3. Proposed Solution to HIS/OpenMI Interoperability

There are many similarities between the HIS and OpenMI. For example, just as the WaterOneFlow web services define a standard interface for describing and accessing data repositories, the OpenMI defines a standard interface for describing and executing models. Likewise, just as the OpenMI includes an object model for communicating data between components, the HIS also includes an object model to communicate time series observations between clients and servers. Despite these similarities, the two technologies were designed independently and therefore have no formal means for interoperability. One of the key goals in this research is to understand how these two technologies can be combined to create an environment able to support both the data and modeling needs of integrated environmental modeling.

3.1. Challenges in Achieving Interoperability

Specific challenges in achieving interoperability between the HIS and OpenMI include inconsistencies in how each system organizes spatiotemporal data, and describes variable and geospatial objects. We found that the most fundamental disconnect is that the HIS is constructed around a time series data model (one location, one variable, many observations through time) while the OpenMI at version 1.4 is constructed around a time slice data model (multiple locations, one variable, one time). This difference is likely a result of the intended purpose for each system. The HIS was built to share observational data (Maidment, 2008), which are typically collected at one monitoring station over a time period. The OpenMI was built to enable model coupling on a time-step basis (Moore and Tindall, 2005). Each model exchanges boundary condition data, which are estimates of a variable at a moment in time over some spatial domain. Overcoming this difference in data organization is possible, but adds complexity to an interoperability solution. OpenMI version 2.0 includes changes that move OpenMI away from a pure model integration standard and closer to a general system integration and workflow environment. We anticipate that this change will simplify the connection between the CUAHSI HIS time series data model and the OpenMI time slice data model, although future implementation work with OpenMI version 2.0 will need to be conducted before drawing any conclusions.

This challenge of differences in how OpenMI and CUAHSI HIS organize spatiotemporal data can be seen in their respective data models. A summarized view of the data models for each system is shown in Figure 3. Although abbreviated, this figure illustrates the most significant concepts that must be translated between the two data models. An OpenMI component is built using the ILinkableComponent interface, which defines the model's computational engine (Gregersen et al., 2007). Furthermore, it must communicate data during a simulation run, such as exchange items, that include element sets, units, and quantities. These OpenMI data exchange objects have similar counterparts in the HIS, although there is not a direct mapping between the two data models. For example, the fundamental OpenMI concept for data communicated among components during simulation is the exchange item (Gregersen et al., 2007). Because of this, it must clearly express not only the data values being transferred, but also metadata including the spatial and variable properties of the values. The most similar HIS concept is the data theme. A data theme is described by a collection of data series which define observations of a specific variable at a specific location over some period of time. However, a theme is not necessarily limited to a single variable, hence a direct mapping between the OpenMI concept of an exchange item and the HIS concept of a data theme will not always be appropriate. There are other examples where there is a clear mapping between HIS and OpenMI concepts. For example, HIS variable and unit objects clearly map to the OpenMI quantity and unit objects. Moreover, an OpenMI element set object can be defined using the HIS sites object. These mappings are summarized in Table 1 and are discussed in more detail in Section 5.

While a direct mapping between all OpenMI and HIS concepts does not exist, it is possible to provide interoperability by making some assumptions. For example, storing model simulation data in the HIS data model is not straight forward, again because the HIS was designed to store time series observations at specific locations. Model simulations usually consist of several data series that differ based on model run. Currently the HIS does not have a formal method for distinguishing between multiple data series having the same variable and site metadata, but differ in terms of their simulation run. This disconnect can temporarily be solved by assuming that each model run can be represented as a different "Method" in ODM terminology (Horsburgh et al., 2008), however this is not a complete solution as the Method is intended to represent data collection methods and not necessary model scenario runs. This example and others like it show that, while the data model presented by the HIS can be expressed in terms of OpenMI objects, it requires some assumptions to do so and ideally would require extention of the HIS database schemas for storing model output data.

3.2. Proposed Approach

Our solution to achieving interoperability between the CUAHSI HIS and the OpenMI is to wrap the HydroDesktop database as an OpenMI-compliant component. This enables the database to serve as a resource to other OpenMI components within a configuration. Two new OpenMI components were developed to achieve this integration: DbReader and DbWriter. The first component, DbReader, searches the HydroDesktop database for time-series data and then translates them into OpenMI exchange item objects that can serve as input to models. By default, these OpenMI exchange items will utilize the HIS controlled vocabulary (Tarboton et al, 2009). The second component, DbWriter, translates one or more OpenMI exchange item objects into time series that can be stored within the HydroDesktop database. Care must be taken to utilize the HIS controlled vocabulary whenever possible in order to remain consistent with other HIS data stored within the HydroDesktop database. By writing data back to the HydroDesktop database, it becomes available to other tools, including map-based and time series-based visualization tools. HydroModeler provides an environment within the HIS architecture enabling loosely integrated modeling capabilities using OpenMI model components, such as the DbReader and DbWriter. The design and implementation of the HydroModeler, DbReader, and DbWriter software are described in Section 4.

3.3. Alternative Designs Considered

Our proposed solution to achieving interoperability between the HIS and OpenMI is the result of a series of alternative approaches that were explored through this research. Our first design was to "wrap" the HIS web services as OpenMI-compliant components. Using this approach, which we named HydroLink, the OpenMI component connected directly to the HIS so that it would retrieve data from the web services whenever data was requested by another component. While an intuitive solution to the problem, the approach was hindered by performance issues because some WaterOneFlow services can take several seconds to return a data request. We believe that this is still a viable solution for some use cases, in particular when real-time data is required by a model or when a web service has been optimized to reduce latency on data requests. However, the hurdles that were encountered suggested that the approach was not ideal for most scenarios.

The next approach explored was to add data caching logic to HydroLink so that the OpenMI component wrapped a directory of time series files stored in the Water Markup Language (WaterML), the format used for data exchange in the CUAHSI HIS (Tarboton et al, 2009). At first, the component was programmed to look for locally cached data when requested by another OpenMI component. If data was not available, it would invoke the HIS web services to automatically download the requested data and cache it for subsequent data requests. The idea was inspired by web browsers which are intelligent about how web pages are requested or cached on client machine, a design feature aimed at providing the most responsive result for end users. Another benefit of the caching approach was that the downloaded directory of WaterML files provided a clear documentation of the input data need to run a particular model. The modeler could easily view these files using other applications and edit them to fill data gaps or replace erroneous values.

While the data caching approach was an adequate technical solution to the problem by combining both the data gathering and data input steps into a single component, the source for information was not always clear as it fed into models. Therefore we felt it necessary to divide the overall workflow of gathering and using data for modeling into three distinct steps: (1) gathering, (2) preparation, and (3) input to models. For the data gathering task, a new tool was created that made batch data requests using the HIS WaterOneFlow web services and downloaded a WaterML file for each request into a local directory. This tool, named FetchWaterML, used a simple CSV file as input to specify a list of time series in the HIS that the user would like to download. The locally stored data could then be pre-processed if necessary, and supplied to models using the HydroLink component.

Our current solution improved on the previous approach by leveraging HydroDesktop for performing the data gathering and data preparation steps. Because HydroDesktop is built on an open source GIS software system, it is able to provide a user-friendly Graphical User Interface (GUI) that better facilitates spatial data searching and visualization. With the introduction of HydroDesktop, the concept of caching WaterML files was replaced with a SQLite database for storing the responses from WaterOneFlow web service calls. This SQLite database is based on the ODM schema and is used to store time-series observations on the user's local machine. The component for reading data from the HIS for input to models, HydroLink, was modified to instead read from the SQLite database behind HydroDesktop. Along with this change in functionality, the HydroLink component was also renamed to DbReader to be more consistent with its role within the HydroDesktop system. HydroModeler was introduced at this time as a plug-in to HydroDesktop to provide an embedded environment for OpenMI model building. Finally, DbWriter was introduced as a means for writing model output data into the SQLite database so that the data can be visualized with HydroDesktop tools.

4. Software Implementation

The HydroModeler was built from the open source OpenMI Editor (OmiED), which is available from the OpenMI Association in the Standard Development Kit (SDK). This editor was modified to integrate with the CUAHSI Hydrologic Information System (HIS) HydroDesktop application via the plug-in interface. HydroDesktop, which was described in Section 2, is the primary client application for the HIS and is aimed at providing a mechanism for discovering, harvesting, and manipulating observation data (Ames et al., 2009). Data is retrieved from HIS WaterOneFlow web services and is stored in a SQLite database repository on the local machine. This local repository is then accessible to any HydroDesktop plug-ins, including HydroModeler, using an Application Programming Interface (API). The HydroModeler relies on the original functionality of the OmiED, such as the ability to build and execute OpenMI model compositions. Reusing this core functionality enabled development efforts to focus on integrating the OpenMI model simulation with the HydroDesktop application.

Two OpenMI components were designed and prototyped with the aim of facilitating the input and output of data between models and the observation database behind HydroDesktop: DbReader and DbWriter. A key step in creating the DbReader component was understanding how and when to extract information from the HydroDesktop database. Likewise, the DbWriter requires a low-level understanding of how and when to extract data from an OpenMI model and write it to the HydroDesktop database. Database reading and writing operations can cause performance issues if not done efficiently, so a key design approach was to ensure that this was done in an efficient matter. The design and implementation for the DbReader and DbWriter components are described following the description of the HydroModeler Graphical User Interface (GUI).

4.1. Graphical User Interface

The HydroModeler Graphical User Interface (GUI) is divided into four main controls: the Browser window, Properties window, Composition window, and the Ribbon toolbar (Figure 4). The Browser window operates similar to a conventional file browser where the user can navigate to find model components or compositions on their local machine. The window automatically filters to show only relevant files: OpenMI-compliant models (*.omi extension) and compositions (*.opr extension). The Properties window automatically populates the metadata for a model component or composition when it is selected from the Browser window. For example, when a composition is selected, the details about the various models that comprise that composition are shown. Furthermore, individual model metadata can be edited and saved directly from the Properties window. Having this functionality embedded within the HydroModeler aids in identifying exchange item mismatches and enables users to modify simulation-based parameters such as start time, end time, and time step.

The Composition window is used to create and execute a linked configuration of model components. Model components or compositions can be added to this window by dragging and dropping them from the Browser window or by using the Ribbon toolbar functionality. Once models have been added to the composition window, the user can establish links between them to create a custom model configuration. Functionality such as linking components is supplied by the underlying OpenMI SDK libraries. The Ribbon toolbar provides a collection of buttons, menus, and dialog boxes for building and running model composition. Its main function is to provide a user friendly and centralized location for the various operations available from HydroModeler.

4.2. DbReader Component

The DbReader component was designed to read observation data from the underlying HydroDesktop database and supply it to a model simulation. To achieve this, the DbReader must be versatile so that it works regardless of the contents of the database. For example, exchange items cannot be predetermined as is typically done for OpenMI model components; instead they are populated from the database during component initialization. Because of this, the DbReader must read data from the HydroDesktop database in two phases. First, it extracts metadata to discover all available exchange items in the database. Then, after a link is connected to one of its output exchange items, it reads the actual time series values into memory. This two step approach reduces the resource footprint by ensuring that extra data series are not loaded into memory. This level of functionality requires the OpenMI ILinkableComponent interface rather than the SDK's IEngine or Simple Model Wrapper (SMW) (Castronova and Goodall, 2010), which are designed for wrapping legacy models and creating process-level components, respectively. Figure 5 illustrates the functionality of the DbReader separated into three parts: the Initialize method, the Add Link method, and the Get Values method.

The Initialize method is called immediately after the component is loaded into a configuration. The DbReader creates output exchange items based on the themes stored in the HydroDesktop database at this time. To do this, data must be extracted and reorganized to conform with the OpenMI exchange item data model. SQL queries are used to obtain theme descriptors for all data series stored in the HydroDesktop database. Using the "ThemeID", additional information is extracted from the local database: "VariableName," "VariableCode," "ThemeName," "ThemeDescription," "SeriesID," "Latitude," "Longitude," "UnitsAbbreviation," "ConversionFactor," "Offset," etc. Finally, this information is mapped to the OpenMI data model to form exchange items (Table 1). These exchange items are then exposed to other components so that linkages can be formed using the HydroModeler composition window controls.

When a link is established between the DbReader and another component, an event is raised. This event calls the Add Link method that first obtains theme information stored on the link. Next, these descriptors are used to query the database for specific data series values. The data values are then stored in a buffer (Oatc.SmartBuffer) along with their corresponding date-times. Finally, this buffer is associated with a specific "LinkID," so that it can be retrieved when values are requested across the corresponding link. This process is repeated every time a link is established between the DbReader and any other component. Once completed, the DbReader is ready for model simulation.

During model simulation, components request values from the DbReader by calling its Get Values method. When this occurs, the correct data buffer is selected using the known "LinkID." The data buffer is then filtered to find the values corresponding to the requested time. If values are found, they are returned to the requesting component. If not, they can be interpolated using OpenMI DataOperations on the known values. Once the appropriate data has been selected, it may also be necessary to perform a spatial interpolation if the input and output element sets are misaligned. To execute a spatial interpolation, the DbReader leverages the Element Mapper class (Oatc.ElementMapper) supplied in the OpenMI SDK. Values are mapped based on a user selected algorithm (nearest neighbor, inverse distance weighting, etc.). Once completed, an array of values are returned to the requesting component. Additionally, unit conversions are performed on-the-fly using auxiliary fields stored in the HydroDesktop Unit Conversions data table. These fields are used to populate the exchange item object (Table 1) so that the OpenMI SDK libraries can be used to automate this process of converting mismatched units between components.

4.3. DbWriter Component

The DbWriter component was developed for saving model simulation results into the HydroDesktop database. The development goal for this component was to seamlessly retrieve data from model components during a simulation run and write them to the underlying HydroDesktop database. Doing so enables modelers to view, edit, and manage simulation results using HydroDesktop plug-in tools. The implementation of this component is divided into four main methods (Figure 6): Initialize, Add Link, Data Changed, and Finish.

During model initialization, the DbWriter must discover what data will be stored in the database and prepare itself for extracting this data during the model simulation. The challenge is that the output exchange items are not known until links have been established. Therefore, the DbWriter component builds a generic input exchange item that can be used to store any component's output data. Additionally, it reads into memory optional metadata fields that are supplied in its *.omi file. These fields represent information that is not available during run time. For example, the modeler is recommended to specify fields for the

HydroDesktop “Source” table, to document who performed the simulation. Furthermore, the HydroDesktop “Method Description” field is used to distinguish between various simulation runs. Lacking such functionality would result in major issues during model calibration. Currently the fields that comprise the HydroDesktop “Method” table, are the only way to distinguish between multiple model simulations in the database, and represent a shortcoming that is addressed in Section 6.

Once the DbWriter and other models are successfully loaded, the user can define links between model outputs and these generic inputs. Every time an output exchange item is linked to the DbWriter, an event is raised that results in the Add Link method being called. The Add Link method performs a series of tasks. First, it subscribes to listener events. These listener events are raised when specific OpenMI methods are called by other components. For example, an event is raised whenever a data exchange is made between two components. By subscribing to these events, the DbWriter can retrieve data values from a component immediately after it completes a time step of simulation. Next, metadata is extracted from the link and is used to define the data theme. This theme information is used to query the database and extract additional information to populate a HydroDesktop data model object. The data model is used to store time series values in a specific structure. It consists of various parameters including variable, time unit, variable unit, measurement method, measurement source, etc. These parameters must be populated carefully to ensure that the resulting data object is compliant with HIS’s controlled vocabulary. Once these parameters have been defined, a site object must be constructed. The HydroDesktop site object consists of many spatial parameters, some of which must be retrieved from the underlying SQLite database. Finally, the data model is stored locally to be used during the run time phase of simulation.

During model simulation, the DbWriter waits for a data exchange to occur. Every data exchange will raise an event which subsequently calls the DbWriter’s Data Changed method. This method first retrieves metadata from the link on which the data transfer occurred. The metadata is used to identify the theme of the data that was transferred. Using this information, the DbWriter requests the values from the component that triggered the event by calling the corresponding Get Values method. This approach allows the DbWriter to retrieve data from components in a non-obtrusive manner. Next, these values are added to their respective data series within the data model object. This information is kept in memory until the model simulation has completed. It is implemented this way to avoid excessive write operations on the database, which can hinder performance.

After model simulation, the Finish method is called to “shutdown” the component. In this phase of simulation, the time-series values stored in the data model object are written to the HydroDesktop database. This is done by first extracting the theme description from the link. This description is then used to check if the database already contains a definition of the theme. If it already exists, then the new values are appended, otherwise a new entry is created. This procedure is continued for every output exchange item connected to the DbWriter.

5. Example Application

A simple yet instructive example is presented in this section to illustrate the use of HydroModeler and the benefit of interoperability between the HIS and OpenMI for integrated environmental modeling. Evapotranspiration (ET) is a hydrological process which relies on observation-based data to define weather conditions. ET describes the loss of water from the land surface to the atmosphere due to evaporation from the surface, including both soil and waterbodies, and transpiration by vegetation (Chow et al., 1988). This section demonstrates how an OpenMI-compliant ET model can utilize CUAHSI HIS input data that is stored in the HydroDesktop, execute in the HydroModeler environment, and then save output results back into the HydroDesktop database.

5.1. ET Model

Evapotranspiration can be approximated by the American Society of Civil Engineering (ASCE) Penman-Monteith (ASCE-PM) approximation. Typical application of this technique consists of first calculating the standardized reference evapotranspiration ET_{sz} (Allen et al., 2005) as

$$ET_{sz} = \frac{\frac{1}{\lambda \rho_w}\Delta(R_n - G) + \gamma \frac{C_n}{T+273} u_2(e_s - e_a)}{\Delta + \gamma(1 + C_d u_2)} \tag{1}$$

where R_n is net radiation, G is soil heat flux density, T is daily averaged temperature, u_2 is daily averaged wind speed, e_s is saturation vapor pressure, e_a is mean vapor pressure, Δ is the saturation vapor pressure-temperature curve, γ is the psychometric constant, λ is the latent heat of vaporization, ρ_w water density, and C_n and C_d are constants. Equation 1 is then multiplied by a crop coefficient (K_c) to estimate potential evapotranspiration (PET). In Equation 1, net radiation (R_n) is expressed as a total of short (S_n) and long (L_n) wave radiation

$$R_n = S_n + L_n \tag{2}$$

where shortwave radiation is calculated using air temperature, date, geographic location, and predetermined coefficients. Similarly, long wave radiation is calculated using air temperature, elevation, and several different coefficients.

5.2. Model Implementation & Application

The ASCE evapotranspiration (ET) model was implemented as two independent OpenMI components using the Simple Model Wrapper (SMW) approach (Castronova and Goodall, 2010). The first component computes the ASCE standardized reference evapotranspiration (ET_{sz}) (Equation 1) and the second computes net solar radiation (R_n) (Equation 2). By implementing the overall process as two components, each one can then be reused for other purposes (e.g. the solar radiation component can be reused in a snow melt model).

The HydroDesktop was used to discover time-series observation data by searching the HIS data repository. The observational data required by the ET model includes temperature, wind speed, dew point temperature, as well as minimum and maximum temperatures. The HydroDesktop manages the download of this data and seamlessly transfers it to the local database repository. Static data required by the model, such as land cover, was downloaded from the United States Geological Survey (USGS) and then used to derive crop coefficients for each gage station. The observation data required by the ET and solar radiation components were supplied by the DbReader across user defined links. Similarly, the DbWriter was used to save simulation results back to the local HydroDesktop data repository. This output data could then be visualized within the HydroDesktop application. The following paragraphs describe how data is translated to and from the model simulation using the DbReader and DbWriter components.

The initialize phase of model simulation is used to "setup" a model component. For the DbReader, this consists of extracting data themes from the HydroDesktop repository, and then creating OpenMI exchange items from them. These exchange items are then supplied to both the ET and solar radiation components via links (Figure 7). Links are used to define the flow of data between components during a simulation. Once a link has been established, the DbReader extracts all data corresponding to the exchange item and stores it in local memory. The DbWriter differs from the DbReader because it initially exposes a generic exchange input item that any component can connect to. Once a connection has been established, the DbWriter uses the link metadata to query the local data repository to find theme information. This theme information is then used to create a HydroDesktop data object.

In the perform time step phase, models communicate across links throughout the simulation. For this model composition, both components require observation data that is stored in a local HydroDesktop repository. Additionally, the ET_{sz} component requires an exchange item provided by the solar radiation component. Model simulation follows a distinct procedure in which all model components advance through simulation on a time step basis; the standard OpenMI procedure. It begins in the "upstream" direction with a data request made by the trigger to the last component in the chain, and continues with subsequent data requests made by each component in an effort to resolve their input data. Once a component is reached that does not rely on an input, the flow of data begins in the "downstream" direction. The flow of data during a model simulation is illustrated in Figure 7. In this case, the DbReader does not rely on input data from another component, therefore the flow of data starts here. The DbReader supplies observation data stored in the local HydroDesktop database to the solar radiation and ET components. Next, the solar radiation component executes its computation and transfers the results to the ET component. The ET component then performs its computation using the input observation data along with the computed solar radiation. After this calculation is complete, the DbWriter is notified that new ET values exist, which are then extracted and stored in a HydroDesktop data object. This routine is repeated for every time step in the simulation.

Finally, after model simulation has completed the components begin their finish method to shutdown. This generally consists of closing input files and releasing allocated memory. The DbWriter, however, performs additional operations during this phase including writing all data series that were collected during simulation to the local HydroDesktop database. The results can then be viewed using the HydroDesktop Graph plug-in (Figure 8). Data values can also be modified using other plug-ins, to remove any outliers or mis-calculations. Using this approach, the local HydroDesktop database functions to store both observation and simulation data. However, further integration between their concepts is necessary for a fully operational data management system.

6. Summary, Discussion, and Future Work

The CUAHSI HIS and OpenMI were developed by different development teams operating in different parts of the world with little communication during formative development years. The potential for synergy between the two systems, however, is clear in that one handles data access and management needs, while the other handles model coupling and integration functionality. Providing interoperability between these two systems is therefore a more complete solution to the challenge of integrated environmental modeling. HydroModeler is one solution to providing interoperability that allows HIS data to serve as input into OpenMI-compliant models and OpenMI-compliant model output to be written to the HydroDesktop database for visualization and analysis. However, as data collection and modeling efforts become more ambitious, issues will undoubtedly continue to arise. Standardization of the approaches for describing environmental data across collection systems and models is required to understand and manage environmental systems.

The scope of integrated environmental modeling is beyond any single group or organization, thus merging standards and approaches will almost certainly be an important part of the process. The work demonstrates that two standards created by two different groups with little formal interaction can still be integrated into a single system. However, it also illustrates that the integration process can be done more seamlessly and completely through establishing overarching standards organizations that ensure that protocols and data standards are synchronized across groups. Both the HIS and OpenMI teams have been working with the Open Geospatial Consortium (OGC) in an effort to establish their own protocols and data exchange standards within the larger body of OGC standards. This work should lead to more universally established standards which are needed to support integrated environmental modeling.

The recent release of the OpenMI version 2.0 introduces several new concepts that we believe will better enable integration of the HIS and OpenMI. These additions generalize the standard by including time span simulation as well as the transfer of generic data types between model components. The new version of the OpenMI standard also includes behind-the-scenes functionality that will aid in the development of calibration routines. These additions greatly enhance the usability of the OpenMI standard across a broader range of research disciplines and offer more flexibility and control over model flow. This new functionality does not directly hinder the work presented here, although all existing OpenMI 1.4 components will have to be upgraded to become OpenMI 2.0 compliant. One potential issue will be converting the DbWriter into an OpenMI 2.0 compliant form, since DbWriter required implementation in a non-standard manner. It is not clear whether this component will need to be redesigned or if it can be converted into an OpenMI 2.0 component using the current design approach. The OpenMI 2.0, like version 1.4, does not include a controlled vocabulary so that work to incorporate semantic integration between the HIS and OpenMI is still relevant and necessary.

Future work will be aimed at adding functionality to the HydroModeler to better accommodate modeling activities. For example, the HydroDesktop database, which was designed primary to accommodate observation data, should be enhanced to better store model related information. There should be extensions to the HydroDesktop database to group time series into time series collections that will map more directly to OpenMI ExchangeItem objects. Furthermore, a more standardized method for storing simulation metadata is required to distinguish between model runs where the output data will be identical in terms of spatial, temporal, and unit representations, but will differ in simulation result. Additional attributes are also necessary to document the differences between model runs, for example the parameters that were changed to create a specific model run. Currently, this is done using the HydroDesktop "Methods" table, however this serves as only a temporary solution. Moreover, the HydroDesktop database contains a "UnitConversions" table to define unit conversions for the data that is supplied to OpenMI components. As of now, these fields must be populated manually. In the future, these fields should populated on-the-fly as data series are downloaded from the HIS.

References

Ahrends, H., Mast, M., Rodgers, C., Kunstmann, H., 2008. Coupled hydrological-economic modelling for optimised irrigated cultivation in a semi-arid catchment of West Africa. Environmental Modelling & Software 23 (4), 385–395.

Allen, R., Environmental, institute (U.S.)., W. R., 2005. The ASCE standardized reference evapotranspiration equation. American Society of Civil Engineers, Reston Va.

Ames, D. P., Horsburgh, J. S., Goodall, J. L., Witeaker, T. L., Tarboton, D. G., Maidment, D. R., Jul. 2009. Introducing the open source CUAHSI Hydrologic Information System desktop application (HIS desktop). In: 18th World IMACS / MODSIM Congress. Cairns, Australia, 4353–4359.

Ames, D. P., Michaelis, C., Anselmo, A., Chen, L., Dunsford, H., 2008. MapWindow GIS. *Encyclopedia of GIS*. Sashi Shekhar and Hui Xiong (Editors). Springer, New York, 633–634.

Argent, R., 2004. An overview of model integration for environmental applications–components, frameworks and semantics. Environmental Modelling & Software 19 (3), 219–234.

Argent, R.M., Perraud, J.M., Rahman, J.M., Grayson, R.B., Podger, G.M., 2009. A new approach to water quality modelling and environmental decision support systems. Environmental Modelling & Software 24 (7), 809–818

Billah, M.M., Goodall, J.L., 2011. Annual and interannual variations in terrestrial water storage during and following a period of drought in South Carolina, USA. Journal of Hydrology 409 (1-2), 472–482.

Bristow, K., Campbell, G., May 1984. On the relationship between incoming solar radiation and daily maximum and minimum temperature? Agricultural and Forest Meteorology 31 (2), 159–166.

Bulatewicz, T., Yang, X., Peterson, J.M., Staggenborg, S., Welch, S.M., Steward, D.R., 2009. Accessible integration of agriculture, groundwater, and economic models using the Open Modeling Interface (OpenMI): methodology and initial results. Hydrology and Earth System Sciences Discussions 6, 7213–7246.

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

Chow, V. T., Maidment, D. R., Mays, L. W., 1988. Applied Hydrology. McGraw-Hill, New York.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S., Apr. 2002. Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6 (2), 86–93.

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

Gregersen, J. B., Gijsbers, P. J. A., Westen, S. J. P., 2007. OpenMI: Open Modelling Interface. Journal of Hydroinformatics 9 (3), 175.

Gruber, T., 2009. Ontology. *Encyclopedia of Database Systems*. Ling Liu and M. Tamer Özsu (Editors). Springer-Verlag, New York.

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

Horsburgh, J. S., Tarboton, D. G., Piasecki, M., Maidment, D. R., Zaslavsky, I., Valentine, D., Whitenack, T., 2009. An integrated system for publishing environmental observations data. Environmental Modelling & Software 24 (8), 879–888.

Huhns, M., Singh, M., 2005. Service-oriented computing: Key concepts and principles. IEEE Internet Computing 9 (1), 75–81.

Kralisch, S., Krause, P., David O., 2005. Using the object modeling system for hydrological model development and application. Advances in Geosciences 4 (1-2), 75–81.

Maidment, D., 1993. Handbook of hydrology. McGraw-Hill, New York.

Maidment, D. R., 2008. Bringing water data together. Journal of Water Resources Planning and Management 134 (2), 95.

Maxwell, R.M., Chow, F.K., Kollet, S.J., 2007. The groundwater-land-surface-atmosphere connection: Soil moisture effects on the atmospheric boundary layer in fully-coupled simulations. Advances in Water Resources 30 (12), 2447–2466

Moore, R. V., Tindall, C., 2005. An overview of the open modelling interface and environment (the OpenMI). Environmental Science & Policy 8 (3), 279–286.

Penman, H. L., Apr. 1948. Natural evaporation from open water, bare soil and grass. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 193 (1032), 120–145.

Syvitski, J., Paola, C., Slingerland, R., Furbish, D., Wiberg, P., Tucker G., 2004. Building a community surface dynamics modeling system: rationale and strategy. *A Report from the Scientific Community to* *the National Science Foundation.*

Tarboton, D.G., Horsburgh, J.S., Maidment, D.R., Whiteaker, T., Zaslavsky, I., Piasecki, M., Goodall, J., Valentine, D., Whitenack, T., 2009. Development of a Community Hydrologic Information System. In: 18th World IMACS / MODSIM Congress. Cairns, Australia, 988–994.

Yu, Z., Pollard, D., Cheng, L., 2006. On continental-scale hydrologic simulations with a coupled hydrologic model. Journal of Hydrology 331 (1-2), 110–124

Table 1: Mapping from the HydroDesktop data definition to the OpenMI data model, to create exchange items.

HydroDesktop / **OpenMI**	*Quantity*	ID	Description	*Unit*	ID	ConversionFactorToSI	OffsetToSI	*ElementSet*	ID	Description	*Element*	ID	XVertex	YVertex
Variables														
VariableName		■												
VariableCode			■											
Units														
UnitsAbbreviation					■									
UnitConversions														
ConversionFactor						■								
Offset							■							
DataThemeDescriptions														
ThemeName									■					
ThemeDescription										■				
Sites														
Longitude													■	
Latitude														■

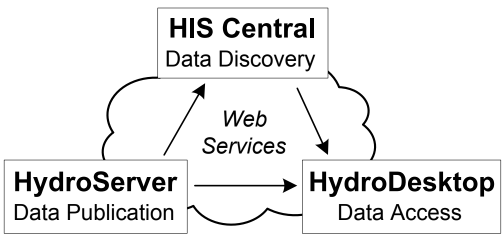

Figure 1: Overview of the CUAHSI Hydrologic Information System

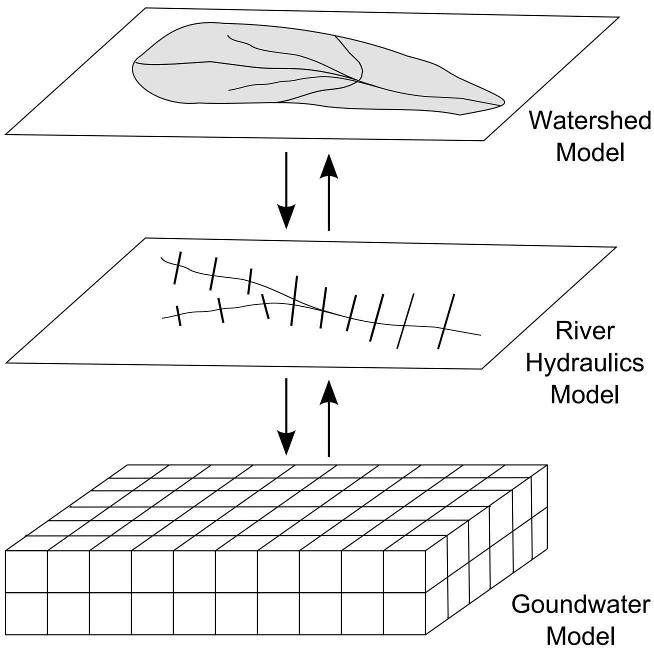

Figure 2: OpenMI defines a standard interface so that models can exchange values during a simulation run. For example, a groundwater model and river hydraulics model could be coupled through the exchange of groundwater heads and river seepage rates.

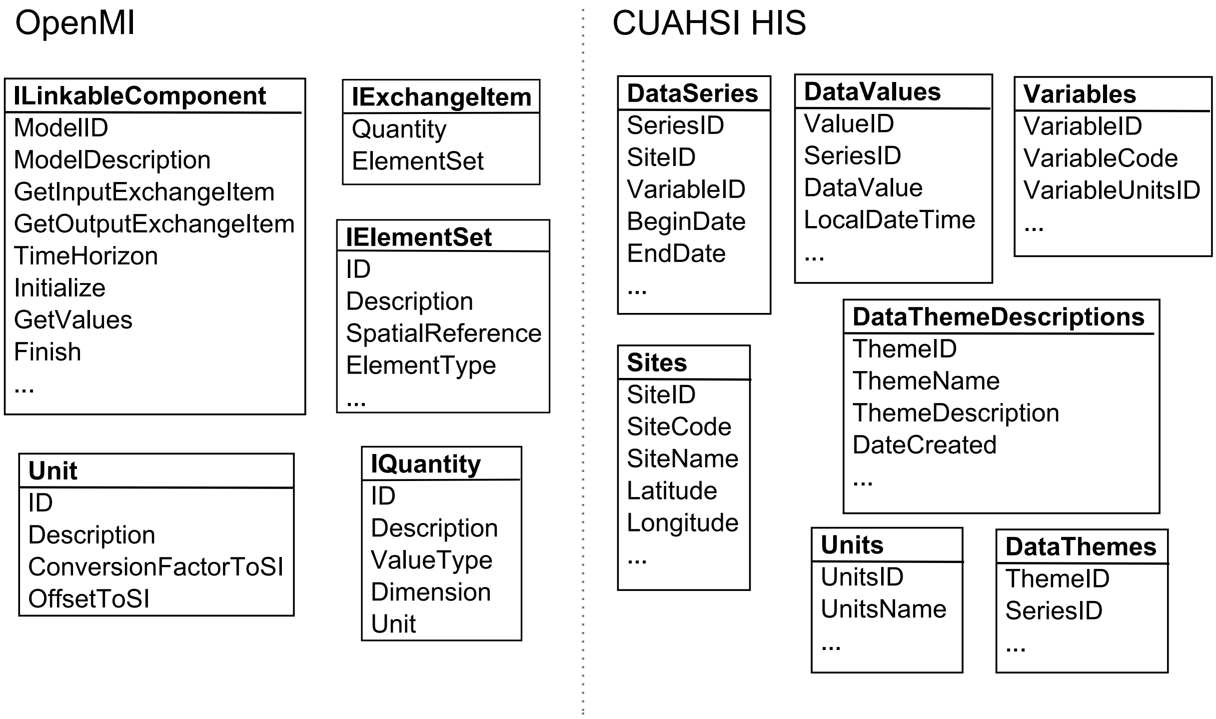

Figure 3: Overview of the common concepts in the OpenMI and CUAHSI HIS data models.

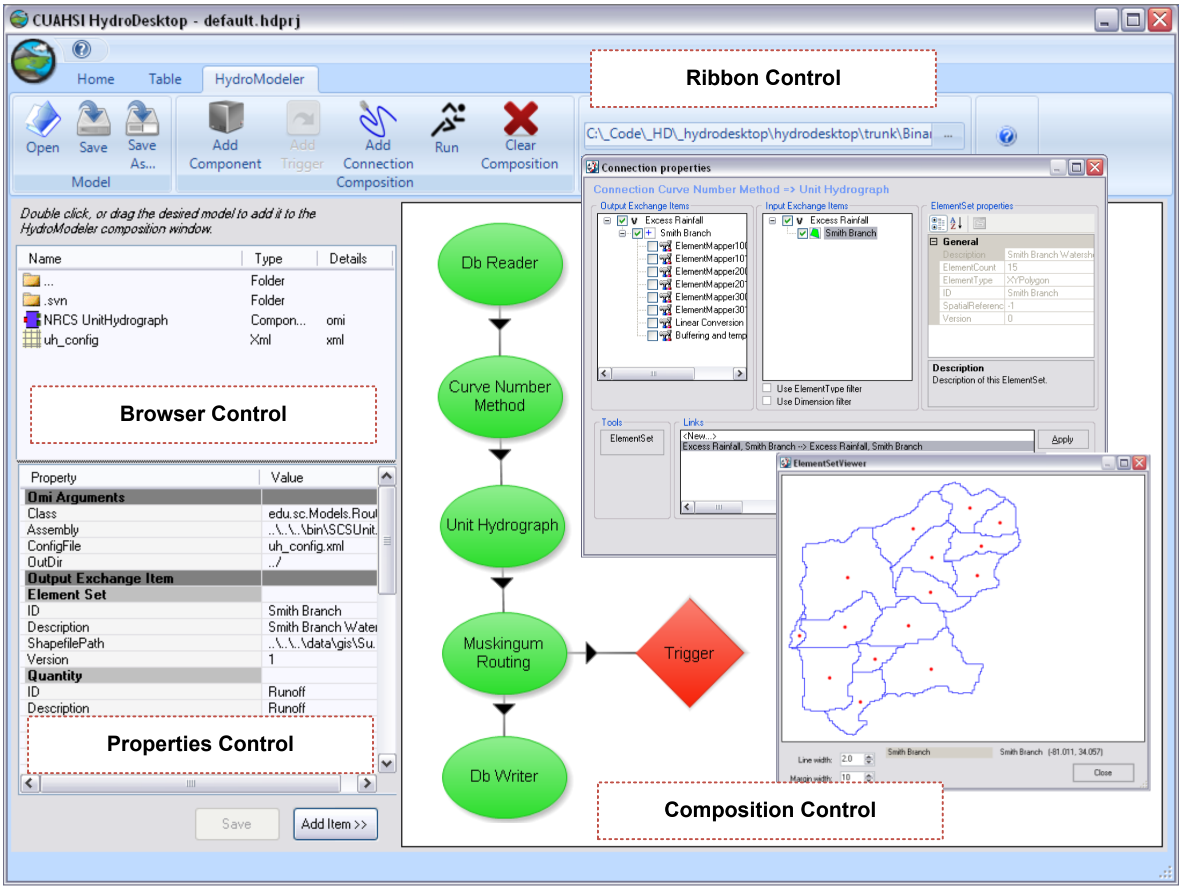

Figure 4: The HydroModeler environment composed of four main software components: the Properties, Browser, Composition, and Ribbon controls.

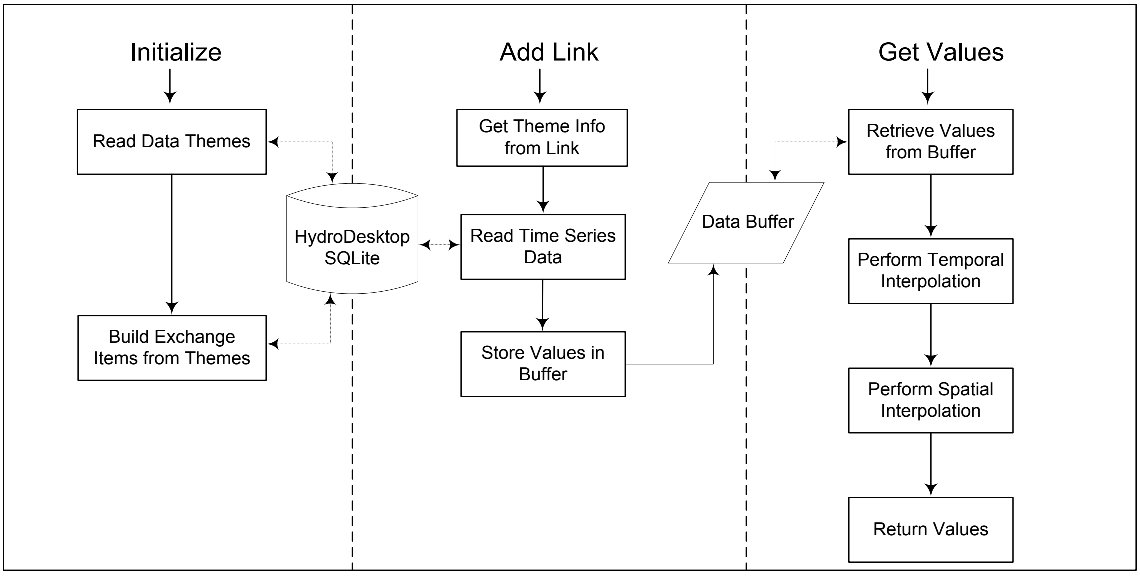

Figure 5: The methodology of the DbReader component separated into three primary methods: Initialize, Add Link, and Get Values.

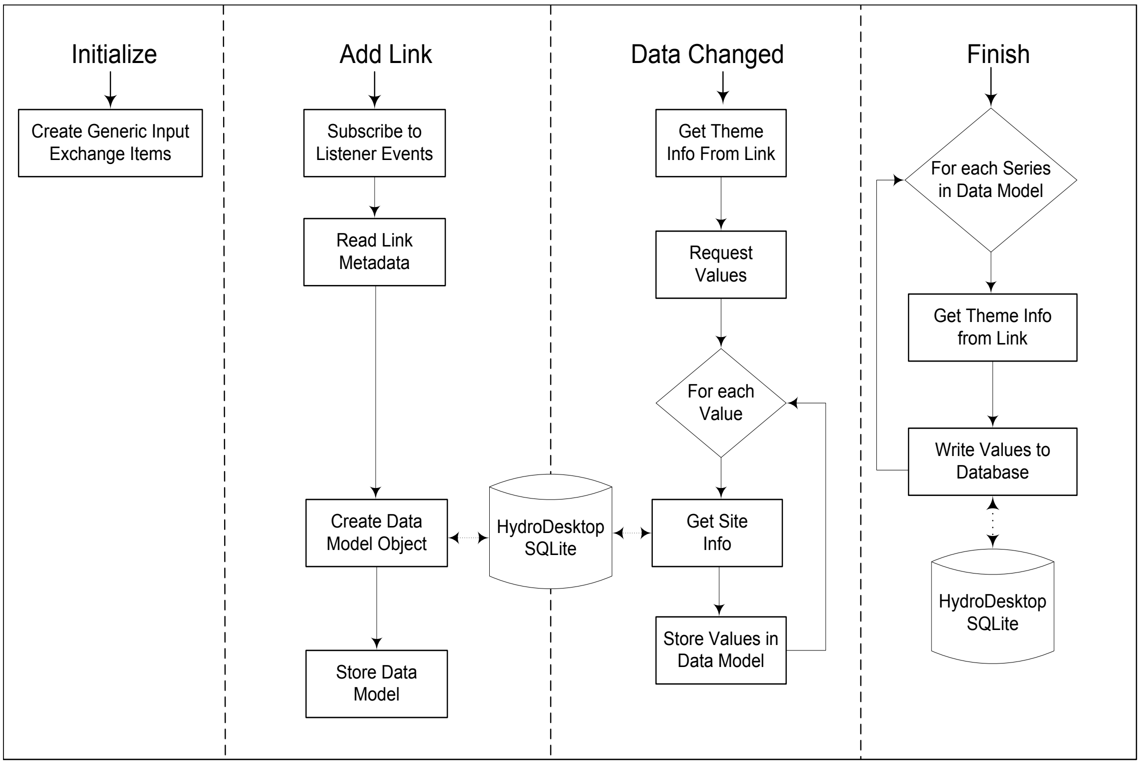

Figure 6: The methodology of the DbWriter component separated into four primary functions: Initialize, Add Link, Data Changed, and Finish.

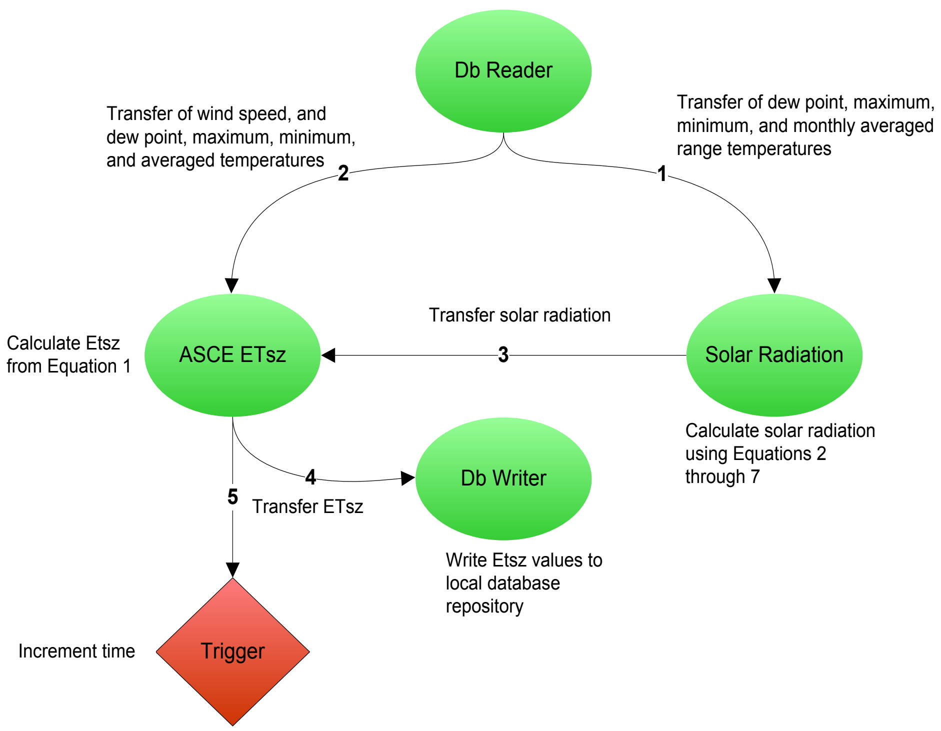

Figure 7: A graphical view of the ET model coupled with the HIS and built using the HydroModeler.

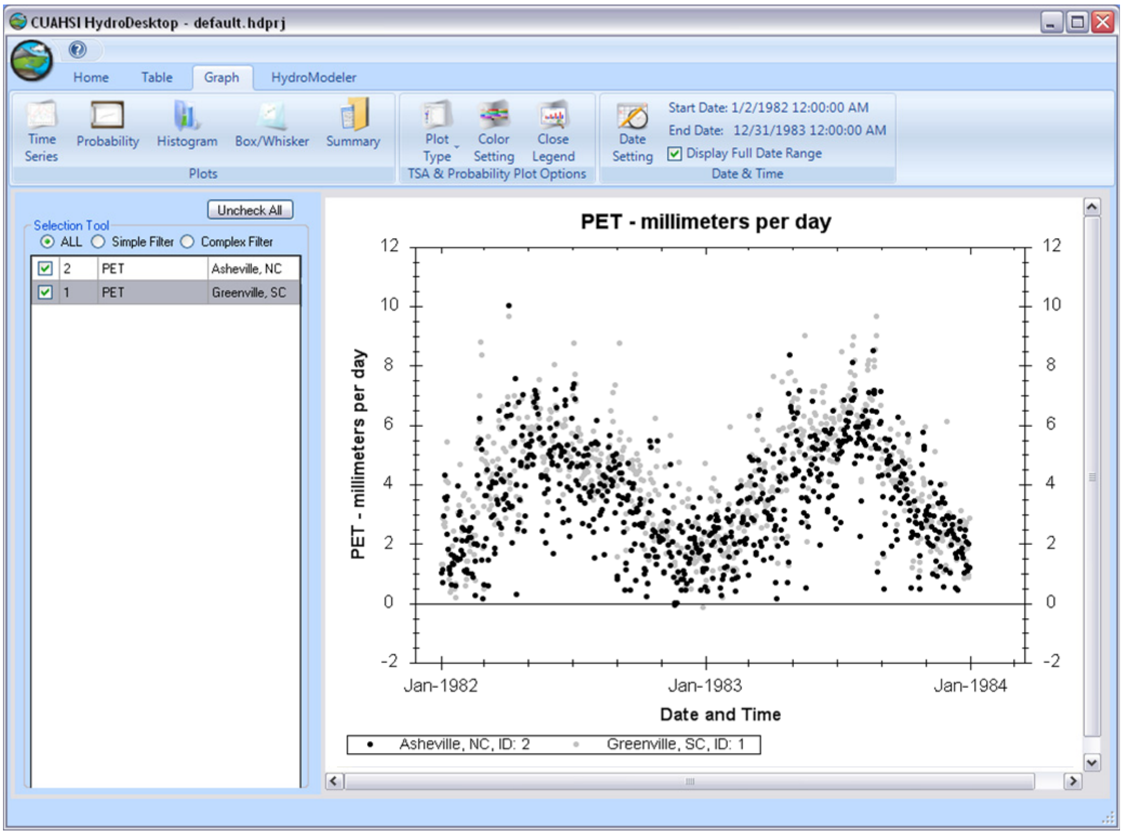

Figure 8: The seasonal trend of daily Potential Evapotranspiration (PET) using weather data from Asheville, NC and Greenville, SC.