

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

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

*^aDepartment 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

25 and the OpenMI can be achieved, and then to use this knowledge to design and prototype
26 a software application that demonstrates system interoperability. The prototype software
27 application, named HydroModeler, is an integrated environmental modeling environment
28 implemented as a plug-in to the CUAHSI HydroDesktop software system (Ames et al.,
29 2009) in order to allow for OpenMI-compliant modeling within the HIS. HydroModeler
30 supports any OpenMI-compliant (Microsoft .NET Framework 4.0) model and enables users
31 to create model configurations where data is supplied by the HIS into simulations and,
32 likewise, data can be written back from a simulation into a local data repository. This data
33 interoperability is possible using two new OpenMI components, a database reader and a
34 database writer. Furthermore, this functionality enables other HydroDesktop tools to work
35 with model output. For example, the HydroDesktop charting and mapping views provide
36 temporal and spatial visualization capabilities for model outputs.

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

46 **2. Background**

47 *2.1. CUAHSI Hydrologic Information System (HIS)*

48 The HIS can be viewed as three separate but interconnected subsystems: HydroServer,
49 HIS Central, and HydroDesktop (Figure 1)(Tarboton et al, 2009). HydroServer is a data
50 sharing tool provided as part of the CUAHSI HIS software stack (Horsburgh et al., 2009).
51 It includes a database schema, known as the Observations Data Model (ODM), for storing

52 observational time series (Horsburgh et al., 2008). In a HydroServer, an ODM database is
53 exposed using the WaterOneFlow web service Application Programming Interface (API),
54 and software tools are provided for managing time series data within an ODM database
55 (Tarboton et al, 2009). HIS Central is a metadata catalog that enables search across dis-
56 tributed HIS data. It includes an ontology and controlled vocabulary to mediate semantic
57 heterogeneity across multiple data providers. In basic terms, the ontology provides the
58 structure needed to integrate disparate systems (i.e. data from different sources) and the
59 controlled vocabulary establishes the precise language needed for inter-system communica-
60 tion (Gruber, 2009). Lastly, the HydroDesktop is a desktop application that enables end
61 users to search, download, and analyze hydrologic data available through HIS Central (Ames
62 et al., 2009). It utilizes a back-end database with a schema similar to the ODM for storing
63 observation data on the user's local machine. The HydroDesktop Graphical User Interface
64 (GUI) is built on an open source Geographic Information System (GIS) platform named
65 MapWindow GIS (Ames et al., 2008) that allows for the extension of core functionality
66 through plug-in software. Plug-in extensions can be developed in the C# (Microsoft .NET
67 Framework 4.0) programming language using a HydroDesktop plug-in interface standard.
68 HydroModeler is one such plug-in extension that adds integrated modeling capabilities to
69 HydroDesktop.

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

81 web services. Moreover, a model can obtain input data directly from a HydroServer, rather
82 than using the graphical HydroDesktop application to prepare input files (e.g. Billah and
83 Goodall, 2011).

84 *2.2. Open Modeling Interface (OpenMI)*

85 The OpenMI is a standard that defines how models exchange data during a simulation
86 run (Moore and Tindall, 2005). It is accompanied by a reference Software Development Kit
87 (SDK) that provides tools for implementing the standard to perform integrated environmen-
88 tal modeling (Gregersen et al., 2007). This research uses OpenMI version 1.4, the current
89 release during the time that the majority of the research was conducted. The OpenMI
90 standard consists of interfaces that can be used to couple models so that they are able to
91 seamlessly exchange data during run time. For example, an integrated modeling effort may
92 require coupling watershed, river hydraulics, and groundwater models, as shown in Figure
93 2. The OpenMI enables such models to be coupled and exchange data necessary to simulate
94 system interactions and dependencies. This approach enables each model to maintain its
95 own identity so that the model can also run independently as well as within a larger sys-
96 tem. Therefore, the OpenMI can be described as a loose integration software architecture
97 (Gregersen et al., 2007) and is in contrast to tight integration approaches where the models
98 are combined into a single system (e.g. Yu et al, 2006; Maxwell et al., 2007; Ahrends et al,
99 2008). Loose integration implies that models are coupled in a “plug-and-play” manner,
100 such that it is possible to reconfigure how they interact without recompiling the source code
101 (Argent, 2004). While the OpenMI was designed to couple large legacy models for envi-
102 ronmental management, it is also possible to create configurations from new components
103 created for research purposes (Bulatewicz et al., 2009; Castronova and Goodall, 2010). One
104 of the most attractive features of component-based modeling is that specific parts of a model
105 system can be interchanged to test their individual impact. This aspect in particular makes
106 the approach useful for scientific research and instruction.

107 The OpenMI concept of a linkable component can be used to couple models, databases,
108 web services, file directories, or any other resource that needs to share data with external

109 components during a model simulation. Authoring and executing a component-based model
110 is mediated by a configuration editor (Gregersen et al., 2007). The OpenMI Association
111 offers a Standard Development Kit (SDK) that includes a basic configuration editor, called
112 the OpenMI Configuration Editor (OmiED). This editor follows the “request-and-reply”
113 communication paradigm defined by the OpenMI standard to achieve system integration
114 (Gregersen et al., 2007). When using a component-based approach, the modeler defines a
115 model configuration that specifies how components within the system are linked together.
116 For example, a forcing variable such as precipitation might be stored within a database and
117 made available to a rainfall/runoff model as a boundary condition, with OmiED orches-
118 trating the data transfer between the database and model. The advantage of modeling a
119 hydrologic system in this manner is that, once the coupling has been defined, each compo-
120 nent can evolve separately from other components within the system as long as the standard
121 interface specification is maintained (Argent, 2004).

122 **3. Proposed Solution to HIS/OpenMI Interoperability**

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

133 *3.1. Challenges in Achieving Interoperability*

134 Specific challenges in achieving interoperability between the HIS and OpenMI include
135 inconsistencies in how each system organizes spatiotemporal data, and describes variable

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

151 This challenge of differences in how OpenMI and CUAHSI HIS organize spatiotemporal
152 data can be seen in their respective data models. A summarized view of the data models
153 for each system is shown in Figure 3. Although abbreviated, this figure illustrates the most
154 significant concepts that must be translated between the two data models. An OpenMI
155 component is built using the ILinkableComponent interface, which defines the model's com-
156 putational engine (Gregersen et al., 2007). Furthermore, it must communicate data during
157 a simulation run, such as exchange items, that include element sets, units, and quantities.
158 These OpenMI data exchange objects have similar counterparts in the HIS, although there is
159 not a direct mapping between the two data models. For example, the fundamental OpenMI
160 concept for data communicated among components during simulation is the exchange item
161 (Gregersen et al., 2007). Because of this, it must clearly express not only the data values be-
162 ing transferred, but also metadata including the spatial and variable properties of the values.
163 The most similar HIS concept is the data theme. A data theme is described by a collection
164 of data series which define observations of a specific variable at a specific location over some

165 period of time. However, a theme is not necessarily limited to a single variable, hence a
166 direct mapping between the OpenMI concept of an exchange item and the HIS concept of a
167 data theme will not always be appropriate. There are other examples where there is a clear
168 mapping between HIS and OpenMI concepts. For example, HIS variable and unit objects
169 clearly map to the OpenMI quantity and unit objects. Moreover, an OpenMI element set
170 object can be defined using the HIS sites object. These mappings are summarized in Table
171 1 and are discussed in more detail in Section 5.

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

185 *3.2. Proposed Approach*

186 Our solution to achieving interoperability between the CUAHSI HIS and the OpenMI
187 is to wrap the HydroDesktop database as an OpenMI-compliant component. This enables
188 the database to serve as a resource to other OpenMI components within a configuration.
189 Two new OpenMI components were developed to achieve this integration: DbReader and
190 DbWriter. The first component, DbReader, searches the HydroDesktop database for time-
191 series data and then translates them into OpenMI exchange item objects that can serve as
192 input to models. By default, these OpenMI exchange items will utilize the HIS controlled

193 vocabulary (Tarboton et al, 2009). The second component, DbWriter, translates one or more
194 OpenMI exchange item objects into time series that can be stored within the HydroDesktop
195 database. Care must be taken to utilize the HIS controlled vocabulary whenever possible in
196 order to remain consistent with other HIS data stored within the HydroDesktop database.
197 By writing data back to the HydroDesktop database, it becomes available to other tools,
198 including map-based and time series-based visualization tools. HydroModeler provides an
199 environment within the HIS architecture enabling loosely integrated modeling capabilities
200 using OpenMI model components, such as the DbReader and DbWriter. The design and
201 implementation of the HydroModeler, DbReader, and DbWriter software are described in
202 Section 4.

203 *3.3. Alternative Designs Considered*

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

215 The next approach explored was to add data caching logic to HydroLink so that the
216 OpenMI component wrapped a directory of time series files stored in the Water Markup
217 Language (WaterML), the format used for data exchange in the CUAHSI HIS (Tarboton
218 et al, 2009). At first, the component was programmed to look for locally cached data when
219 requested by another OpenMI component. If data was not available, it would invoke the
220 HIS web services to automatically download the requested data and cache it for subsequent

221 data requests. The idea was inspired by web browsers which are intelligent about how web
222 pages are requested or cached on client machine, a design feature aimed at providing the
223 most responsive result for end users. Another benefit of the caching approach was that
224 the downloaded directory of WaterML files provided a clear documentation of the input
225 data need to run a particular model. The modeler could easily view these files using other
226 applications and edit them to fill data gaps or replace erroneous values.

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

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

250 a means for writing model output data into the SQLite database so that the data can be
251 visualized with HydroDesktop tools.

252 **4. Software Implementation**

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

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

276 *4.1. Graphical User Interface*

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

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

298 *4.2. DbReader Component*

299 The DbReader component was designed to read observation data from the underlying
300 HydroDesktop database and supply it to a model simulation. To achieve this, the DbReader
301 must be versatile so that it works regardless of the contents of the database. For example,
302 exchange items cannot be predetermined as is typically done for OpenMI model components;
303 instead they are populated from the database during component initialization. Because of

304 this, the DbReader must read data from the HydroDesktop database in two phases. First,
305 it extracts metadata to discover all available exchange items in the database. Then, after a
306 link is connected to one of its output exchange items, it reads the actual time series values
307 into memory. This two step approach reduces the resource footprint by ensuring that extra
308 data series are not loaded into memory. This level of functionality requires the OpenMI
309 ILinkableComponent interface rather than the SDK's IEngine or Simple Model Wrapper
310 (SMW) (Castronova and Goodall, 2010), which are designed for wrapping legacy models
311 and creating process-level components, respectively. Figure 5 illustrates the functionality of
312 the DbReader separated into three parts: the Initialize method, the Add Link method, and
313 the Get Values method.

314 The Initialize method is called immediately after the component is loaded into a con-
315 figuration. The DbReader creates output exchange items based on the themes stored in
316 the HydroDesktop database at this time. To do this, data must be extracted and reor-
317 ganized to conform with the OpenMI exchange item data model. SQL queries are used
318 to obtain theme descriptors for all data series stored in the HydroDesktop database. Us-
319 ing the "ThemeID", additional information is extracted from the local database: "Vari-
320 ableName," "VariableCode," "ThemeName," "ThemeDescription," "SeriesID," "Latitude,"
321 "Longitude," "UnitsAbbreviation," "ConversionFactor," "Offset," etc. Finally, this infor-
322 mation is mapped to the OpenMI data model to form exchange items (Table 1). These
323 exchange items are then exposed to other components so that linkages can be formed using
324 the HydroModeler composition window controls.

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

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

347 *4.3. DbWriter Component*

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

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

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

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

385 During model simulation, the DbWriter waits for a data exchange to occur. Every data
386 exchange will raise an event which subsequently calls the DbWriter’s Data Changed method.
387 This method first retrieves metadata from the link on which the data transfer occurred.
388 The metadata is used to identify the theme of the data that was transferred. Using this
389 information, the DbWriter requests the values from the component that triggered the event

390 by calling the corresponding Get Values method. This approach allows the DbWriter to
391 retrieve data from components in a non-obtrusive manner. Next, these values are added
392 to their respective data series within the data model object. This information is kept in
393 memory until the model simulation has completed. It is implemented this way to avoid
394 excessive write operations on the database, which can hinder performance.

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

402 **5. Example Application**

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

412 *5.1. ET Model*

413 Evapotranspiration can be approximated by the American Society of Civil Engineering
414 (ASCE) Penman-Monteith (ASCE-PM) approximation. Typical application of this tech-
415 nique consists of first calculating the standardized reference evapotranspiration ET_{sz} (Allen

416 et al., 2005) as

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

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

$$R_n = S_n + L_n \quad (2)$$

424 where shortwave radiation is calculated using air temperature, date, geographic location,
425 and predetermined coefficients. Similarly, long wave radiation is calculated using air tem-
426 perature, elevation, and several different coefficients.

427 5.2. Model Implementation & Application

428 The ASCE evapotranspiration (ET) model was implemented as two independent OpenMI
429 components using the Simple Model Wrapper (SMW) approach (Castronova and Goodall,
430 2010). The first component computes the ASCE standardized reference evapotranspiration
431 (ET_{sz}) (Equation 1) and the second computes net solar radiation (R_n) (Equation 2). By
432 implementing the overall process as two components, each one can then be reused for other
433 purposes (e.g. the solar radiation component can be reused in a snow melt model).

434 The HydroDesktop was used to discover time-series observation data by searching the
435 HIS data repository. The observational data required by the ET model includes temperature,
436 wind speed, dew point temperature, as well as minimum and maximum temperatures. The
437 HydroDesktop manages the download of this data and seamlessly transfers it to the local
438 database repository. Static data required by the model, such as land cover, was downloaded
439 from the United States Geological Survey (USGS) and then used to derive crop coefficients
440 for each gage station. The observation data required by the ET and solar radiation com-
441 ponents were supplied by the DbReader across user defined links. Similarly, the DbWriter

442 was used to save simulation results back to the local HydroDesktop data repository. This
443 output data could then be visualized within the HydroDesktop application. The following
444 paragraphs describe how data is translated to and from the model simulation using the
445 DbReader and DbWriter components.

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

457 In the perform time step phase, models communicate across links throughout the simu-
458 lation. For this model composition, both components require observation data that is stored
459 in a local HydroDesktop repository. Additionally, the ET_{sz} component requires an exchange
460 item provided by the solar radiation component. Model simulation follows a distinct pro-
461 cedure in which all model components advance through simulation on a time step basis;
462 the standard OpenMI procedure. It begins in the “upstream” direction with a data request
463 made by the trigger to the last component in the chain, and continues with subsequent data
464 requests made by each component in an effort to resolve their input data. Once a compo-
465 nent is reached that does not rely on an input, the flow of data begins in the “downstream”
466 direction. The flow of data during a model simulation is illustrated in Figure 7. In this case,
467 the DbReader does not rely on input data from another component, therefore the flow of
468 data starts here. The DbReader supplies observation data stored in the local HydroDesktop
469 database to the solar radiation and ET components. Next, the solar radiation component
470 executes its computation and transfers the results to the ET component. The ET component

471 then performs its computation using the input observation data along with the computed
472 solar radiation. After this calculation is complete, the DbWriter is notified that new ET
473 values exist, which are then extracted and stored in a HydroDesktop data object. This
474 routine is repeated for every time step in the simulation.

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

484 **6. Summary, Discussion, and Future Work**

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

497 The scope of integrated environmental modeling is beyond any single group or organi-
498 zation, thus merging standards and approaches will almost certainly be an important part

499 of the process. The work demonstrates that two standards created by two different groups
500 with little formal interaction can still be integrated into a single system. However, it also
501 illustrates that the integration process can be done more seamlessly and completely through
502 establishing overarching standards organizations that ensure that protocols and data stan-
503 dards are synchronized across groups. Both the HIS and OpenMI teams have been working
504 with the Open Geospatial Consortium (OGC) in an effort to establish their own protocols
505 and data exchange standards within the larger body of OGC standards. This work should
506 lead to more universally established standards which are needed to support integrated en-
507 vironmental modeling.

508 The recent release of the OpenMI version 2.0 introduces several new concepts that we
509 believe will better enable integration of the HIS and OpenMI. These additions generalize
510 the standard by including time span simulation as well as the transfer of generic data types
511 between model components. The new version of the OpenMI standard also includes behind-
512 the-scenes functionality that will aid in the development of calibration routines. These
513 additions greatly enhance the usability of the OpenMI standard across a broader range
514 of research disciplines and offer more flexibility and control over model flow. This new
515 functionality does not directly hinder the work presented here, although all existing OpenMI
516 1.4 components will have to be upgraded to become OpenMI 2.0 compliant. One potential
517 issue will be converting the DbWriter into an OpenMI 2.0 compliant form, since DbWriter
518 required implementation in a non-standard manner. It is not clear whether this component
519 will need to be redesigned or if it can be converted into an OpenMI 2.0 component using the
520 current design approach. The OpenMI 2.0, like version 1.4, does not include a controlled
521 vocabulary so that work to incorporate semantic integration between the HIS and OpenMI
522 is still relevant and necessary.

523 Future work will be aimed at adding functionality to the HydroModeler to better accom-
524 modate modeling activities. For example, the HydroDesktop database, which was designed
525 primary to accommodate observation data, should be enhanced to better store model re-
526 lated information. There should be extensions to the HydroDesktop database to group time
527 series into time series collections that will map more directly to OpenMI ExchangeItem

528 objects. Furthermore, a more standardized method for storing simulation metadata is re-
529 quired to distinguish between model runs where the output data will be identical in terms of
530 spatial, temporal, and unit representations, but will differ in simulation result. Additional
531 attributes are also necessary to document the differences between model runs, for example
532 the parameters that were changed to create a specific model run. Currently, this is done
533 using the HydroDesktop “Methods” table, however this serves as only a temporary solution.
534 Moreover, the HydroDesktop database contains a “UnitConversions” table to define unit
535 conversions for the data that is supplied to OpenMI components. As of now, these fields
536 must be populated manually. In the future, these fields should be populated on-the-fly as data
537 series are downloaded from the HIS.

538 References

- 539 Ahrends, H., Mast, M., Rodgers, C., Kunstmann, H., 2008. Coupled hydrological-economic modelling for
540 optimised irrigated cultivation in a semi-arid catchment of West Africa. *Environmental Modelling &*
541 *Software* 23 (4), 385–395.
- 542 Allen, R., Environmental, institute (U.S.), W. R., 2005. The ASCE standardized reference evapotranspira-
543 tion equation. American Society of Civil Engineers, Reston Va.
- 544 Ames, D. P., Horsburgh, J. S., Goodall, J. L., Witeaker, T. L., Tarboton, D. G., Maidment, D. R., Jul. 2009.
545 Introducing the open source CUAHSI Hydrologic Information System desktop application (HIS desktop).
546 In: 18th World IMACS / MODSIM Congress. Cairns, Australia, 4353–4359.
- 547 Ames, D. P., Michaelis, C., Anselmo, A., Chen, L., Dunsford, H., 2008. MapWindow GIS. *Encyclopedia of*
548 *GIS*. Sashi Shekhar and Hui Xiong (Editors). Springer, New York, 633–634.
- 549 Argent, R., 2004. An overview of model integration for environmental applications—components, frameworks
550 and semantics. *Environmental Modelling & Software* 19 (3), 219–234.
- 551 Argent, R.M., Perraud, J.M., Rahman, J.M., Grayson, R.B., Podger, G.M., 2009. A new approach to water
552 quality modelling and environmental decision support systems. *Environmental Modelling & Software*
553 24 (7), 809–818
- 554 Billah, M.M., Goodall, J.L., 2011. Annual and interannual variations in terrestrial water storage during and
555 following a period of drought in South Carolina, USA. *Journal of Hydrology* 409 (1-2), 472–482.
- 556 Bristow, K., Campbell, G., May 1984. On the relationship between incoming solar radiation and daily
557 maximum and minimum temperature? *Agricultural and Forest Meteorology* 31 (2), 159–166.
- 558 Bulatewicz, T., Yang, X., Peterson, J.M., Staggenborg, S., Welch, S.M., Steward, D.R., 2009. Accessible inte-

559 gration of agriculture, groundwater, and economic models using the Open Modeling Interface (OpenMI):
560 methodology and initial results. *Hydrology and Earth System Sciences Discussions* 6, 7213–7246.

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

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

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

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

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

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

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

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

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

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

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

582 Maidment, D. R., 2008. Bringing water data together. *Journal of Water Resources Planning and Management*
583 134 (2), 95.

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

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

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

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

593 *the National Science Foundation.*

594 Tarboton, D.G., Horsburgh, J.S., Maidment, D.R., Whiteaker, T., Zaslavsky, I., Piasecki, M., Goodall, J.,
595 Valentine, D., Whitenack, T., 2009. Development of a Community Hydrologic Information System. In:
596 18th World IMACS / MODSIM Congress. Cairns, Australia, 988–994.

597 Yu, Z., Pollard, D., Cheng, L., 2006. On continental-scale hydrologic simulations with a coupled hydrologic
598 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	<i>Quantity</i> ID Description	<i>Unit</i> ID ConversionFactorToSI OffsetToSI	<i>ElementSet</i> ID Description	<i>Element</i> ID XVertex YVertex
<i>Variables</i> VariableName VariableCode		■ ■			
<i>Units</i> UnitsAbbreviation			■		
<i>UnitConversions</i> ConversionFactor Offset			■ ■		
<i>DataThemeDescriptions</i> ThemeName ThemeDescription				■ ■	
<i>Sites</i> Longitude Latitude					■ ■

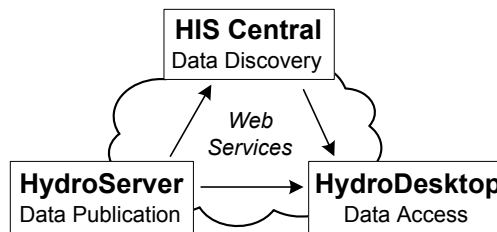


Figure 1: Overview of the CUAHSI Hydrologic Information System

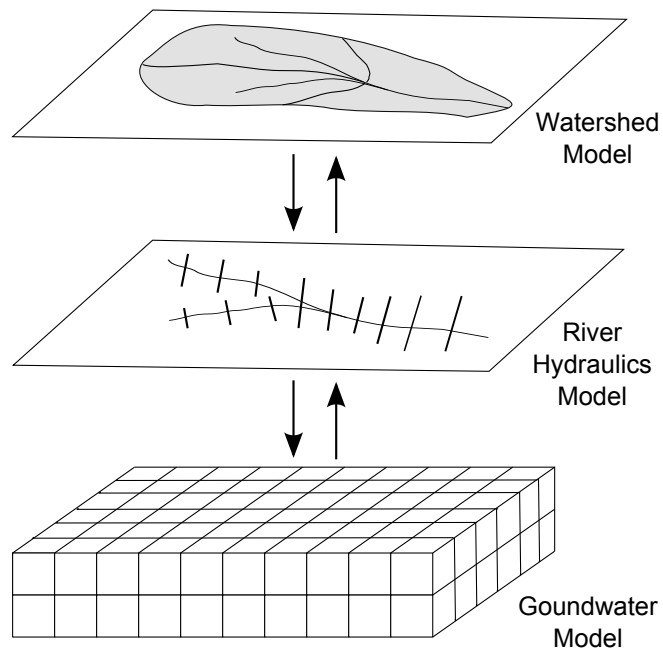
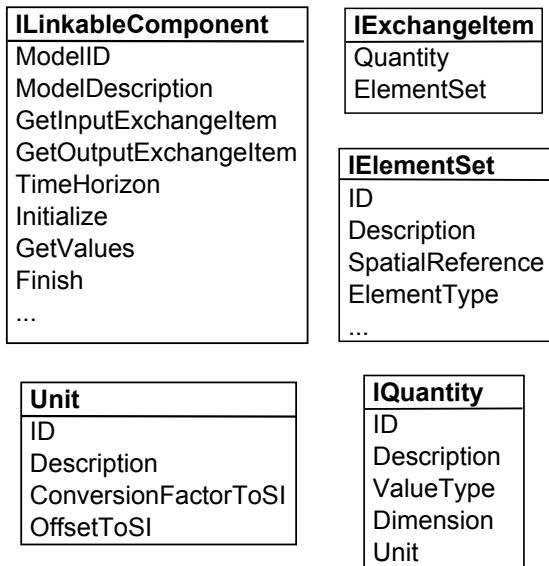


Figure 2: OpenMI defines a standard interface so that models can exchange values during a simulation run. For example, a groundwater model and river hydraulics model could be coupled through the exchange of groundwater heads and river seepage rates.

OpenMI



CUAHSI HIS

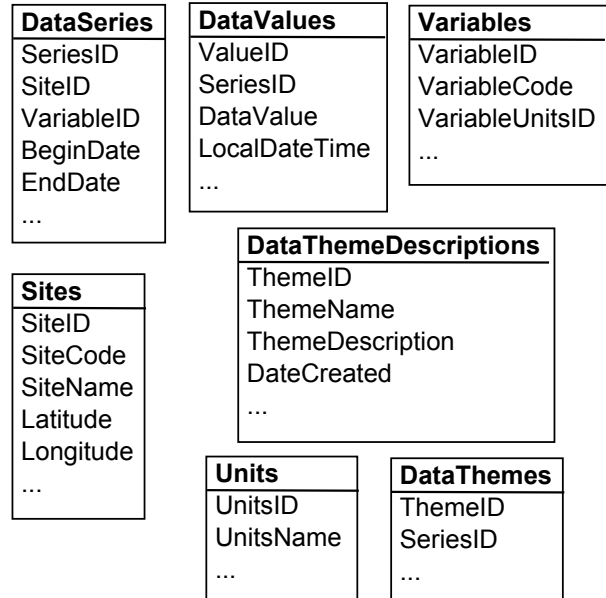


Figure 3: Overview of the common concepts in the OpenMI and CUAHSI HIS data models.

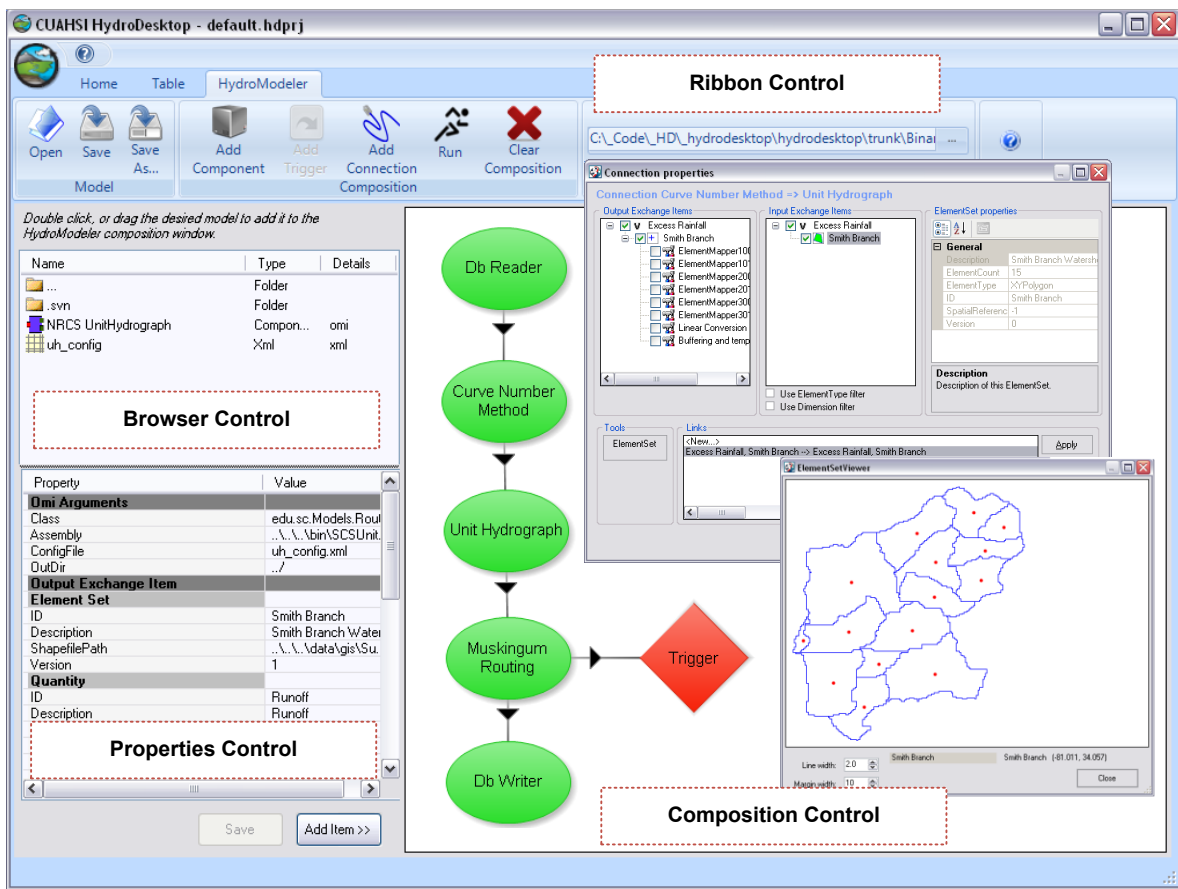


Figure 4: The HydroModeler environment composed of four main software components: the Properties, Browser, Composition, and Ribbon controls.

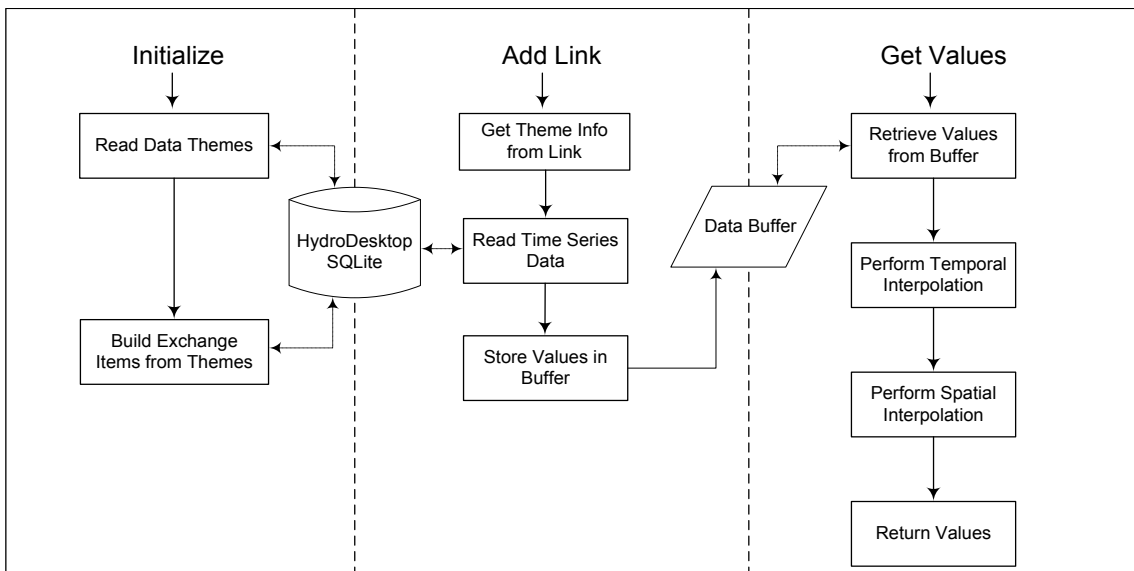


Figure 5: The methodology of the DbReader component separated into three primary methods: Initialize, Add Link, and Get Values.

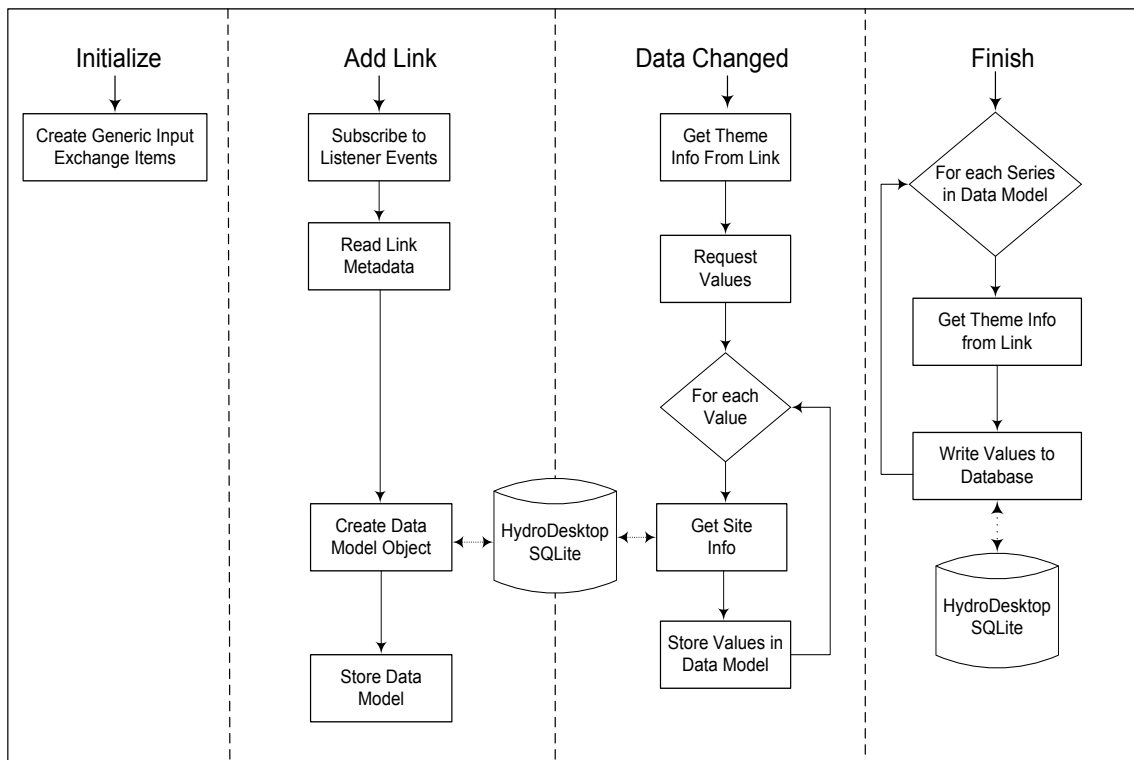


Figure 6: The methodology of the DbWriter component separated into four primary functions: Initialize, Add Link, Data Changed, and Finish.

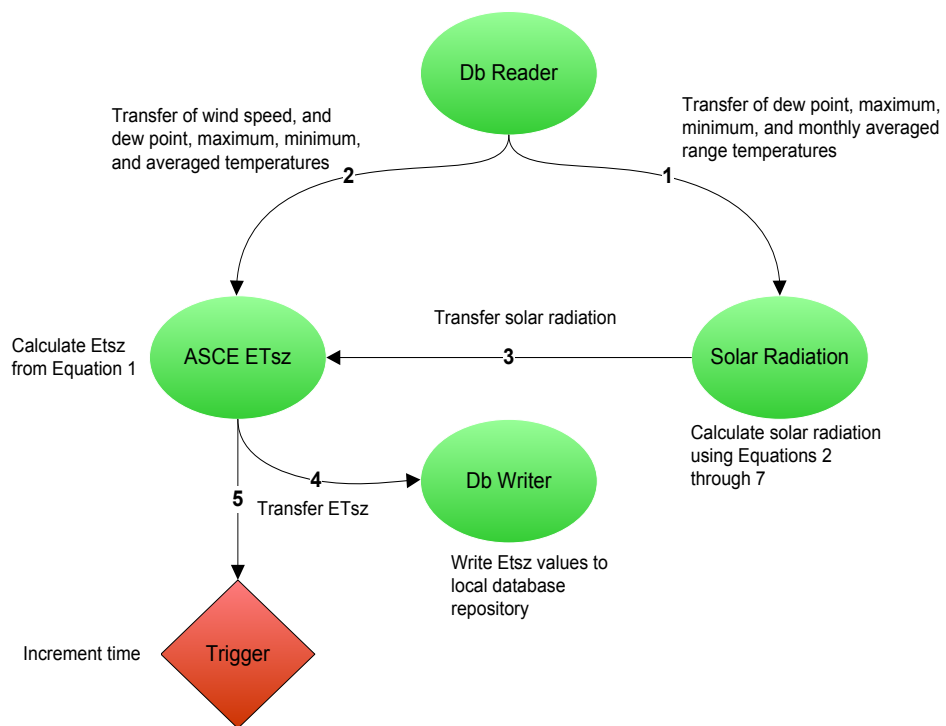


Figure 7: A graphical view of the ET model coupled with the HIS and built using the HydroModeler.

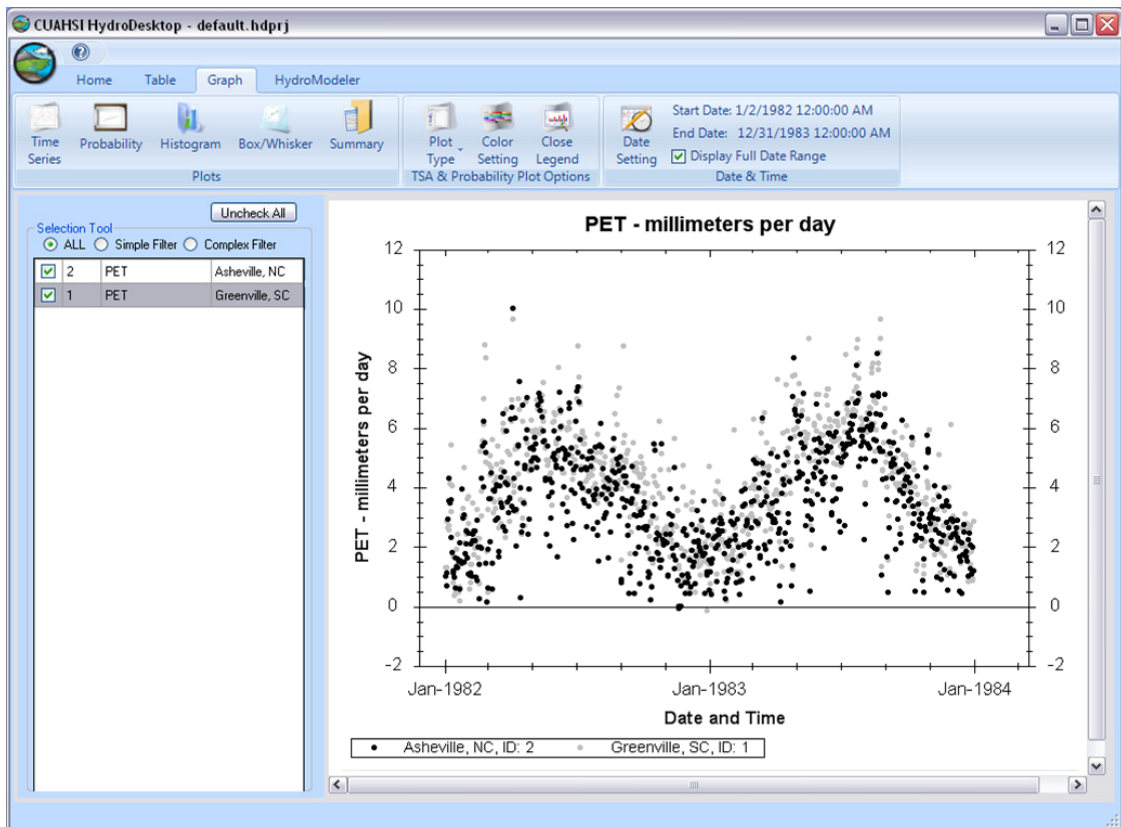


Figure 8: The seasonal trend of daily Potential Evapotranspiration (PET) using weather data from Asheville, NC and Greenville, SC.