

# 1 **Design of a Metadata Framework for Hydrologic Models with an Example** 2 **Application in HydroShare**

3 Mohamed M. Morsy<sup>a,b</sup>, Jonathan L. Goodall<sup>a\*</sup>, Anthony M. Castronova<sup>c</sup>, Pabitra Dash<sup>c</sup>,  
4 Venkatesh Merwade<sup>d</sup>, Jeffrey M. Sadler<sup>a</sup>, Md. Adnan Rajib<sup>d</sup>, Jeffery S. Horsburgh<sup>c</sup>, David G.  
5 Tarboton<sup>c</sup>

6  
7 <sup>a</sup> Department of Civil and Environmental Engineering, University of Virginia, 351 McCormick  
8 Road, PO Box 400742, Charlottesville, VA, 22908, USA

9 <sup>b</sup> Irrigation and Hydraulics Department, Faculty of Engineering, Cairo University, P.O. Box  
10 12211, Giza 12613, Egypt

11 <sup>c</sup> Utah Water Research Laboratory, Utah State University, 8200 Old Main Hill, Logan, UT  
12 84322-8200, USA

13 <sup>d</sup> School of Civil Engineering, Purdue University, 550 Stadium Mall Drive, West Lafayette, IN  
14 47907, USA

15

## 16 *Highlights:*

- 17 • The design of a metadata framework for model programs and instances is presented
- 18 • The metadata framework is implemented within the HydroShare system
- 19 • The framework is built from general standards like RDF, Dublin Core, and BagIt
- 20 • The implementation is demonstrated for a hydrologic model publication use case
- 21 • The implementation assists in model sharing, publication, reuse and reproducibility

22

23 **Abstract**

24 Hydrologists rely on a variety of computational models to make predictions, test hypotheses, and  
25 address specific problems related to hydrologic science and water resources management.  
26 Scientists and engineers must devote significant effort preparing these computational models.  
27 While significant attention has been devoted to sharing and reusing hydrologic data, less  
28 attention has been devoted to sharing and reusing hydrologic models. A first step toward  
29 increasing hydrologic model sharing and reuse is to define a general metadata framework for  
30 models that is flexible and, therefore, applicable across the wide variety of models used by  
31 hydrologists. To this end, this paper proposes a general approach for representing hydrologic  
32 model metadata that extends the Dublin Core metadata framework. The framework is  
33 implemented within the HydroShare system and applied for a model sharing use case. This  
34 example application demonstrates how the metadata framework implemented within HydroShare  
35 can assist in model sharing, publication, reuse, and reproducibility.

36

37 *Keywords:*

38 hydrologic modeling; model metadata; linked data; Dublin Core metadata initiative;  
39 reproducibility

40

41 *Software Availability:*

42 The software created in this research is available free and open source as part of the  
43 larger HydroShare software repository. The HydroShare software repository is managed  
44 through GitHub and is available at <https://github.com/hydroshare/hydroshare>.

45

## 46 **1. Introduction**

47           A large variety of hydrologic models exists, with each model tailored to address specific  
48 challenges related to hydrologic science and water resources management (Singh et al., 2002;  
49 2006). These models have grown in complexity, with many simulating increasingly detailed  
50 processes occurring within water systems. When scientists and engineers use models, they must  
51 devote significant effort to collect data, construct model inputs, and calibrate and validate model  
52 parameters. Many hydrologic models also require sophisticated data pre-processing routines,  
53 often with many manual steps (e.g., Billah et al., 2016). For this reason, many models come with  
54 supporting applications such as Geographic Information System (GIS) interfaces, calibration  
55 tools, visualization software, and other utility software systems to assist in the data preparation  
56 process (e.g., Winchell et al., 2007). These data pre-processing steps must be repeated each time  
57 a new model is created to simulate a system. This introduces a number of challenges. From a  
58 pragmatic perspective, it is an inefficient use of scientists' time. Perhaps more importantly, it  
59 inhibits scientists' ability to reproduce studies that have a significant computational modeling  
60 component (David et al., 2016; Essawy et al., 2016; Gil et al., 2016).

61           One way to begin to address these challenges is through better approaches for sharing and  
62 reusing models built by others. Just as there has been a major push to make better use of data  
63 collected and maintained by others, the scientific community can benefit from a similar push to  
64 make better use of models built by others. Data sharing and reuse has been strengthened through  
65 the adoption of agreed on metadata frameworks. Geospatial data, in particular, has benefited  
66 from widely used metadata frameworks that allow scientists and engineers to more easily reuse  
67 data collected by others (e.g., ISO, 2003; 2011). More recently, hydrologic time series data have  
68 also benefited from the adoption of commonly used metadata frameworks (e.g., Taylor, 2014).

69 While many metadata frameworks exist, none specifically addresses computational hydrologic  
70 models. Thus, the objective of this research was to design and implement such a metadata  
71 framework for hydrologic models.

72 Designing a metadata framework for hydrologic models poses unique challenges  
73 compared to other data types. First, the data required for models are heterogeneous and, in the  
74 case of hydrologic models, input for a single simulation can include dozens, if not hundreds, of  
75 data files. These files describe properties of the modeling elements, parameters, forcing  
76 functions, boundary conditions, and other data needed to execute the model for a given system.  
77 Each model largely adopts its own structure and semantics for storing data, making it difficult to  
78 standardize across models. Second, hydrologists make use of a large and diverse set of  
79 computational models; Singh (2002) cataloged over 65 hydrologic models focusing on watershed  
80 hydrology alone. Hydrologists will likely continue to make use of a broad range of models,  
81 because each model is tailored for a given application. Some models are well suited for  
82 urbanized watersheds, while others are better suited for agricultural watersheds; some models are  
83 best for droughts, others for floods; some target regional-scale systems, others plot or hill-slope  
84 scales. Each model adopts unique data structures and semantics for both input and output data. A  
85 model metadata framework, therefore, must not force all models into a fixed structure, but rather  
86 be flexible and able to accommodate this diversity of models.

87 Some studies have begun to address the problem of designing a metadata framework for  
88 computational models. The Content Standard for Computational Models (Hill et al., 2001) was  
89 one of the first attempts at providing detailed metadata about a numerical model that includes the  
90 input and output data for model scenarios. Wosniok and Lehfeldt (2013) provide a concept for  
91 metadata-driven architecture for computational fluid dynamics simulations and a way to

92 integrate model descriptions into spatial data infrastructures. The Community Surface Dynamics  
93 Modeling System (CSDMS) created a metadata framework and used it to describe over 180  
94 geoscience models, including over 50 hydrologic models within its model catalog (see  
95 <http://csdms.colorado.edu>). The CSDMS model category focuses on the software for executing a  
96 model, what we refer to in this paper as a *model program*. It does not extend to the input files for  
97 a specific model simulation, or what we refer to in this paper as a *model instance*. The metadata  
98 included in CSDMS also do not follow higher-level metadata standards like Dublin Core.

99       Much of the past research on model metadata has focused on component-based modeling  
100 systems. Component-based modeling systems are a tool for integrated environmental modeling  
101 where model applications are constructed from a set of “plug-and-play” model components that  
102 can be interchanged for different applications (Argent, 2004; Laniak et al., 2013). Metadata  
103 frameworks have been proposed for model components generally (Elag and Goodall, 2013), the  
104 component interfaces (Gregersen et al., 2007; Peckham et al., 2013), and the variables passed  
105 between linked components (Peckham, 2014). Our work is different in that we focus on  
106 standalone model programs instead of component-based modeling systems. We take this focus  
107 because, while the adoption of component-based modeling systems is growing, the vast majority  
108 of ongoing hydrologic studies are using standalone model applications and a metadata  
109 framework is needed to enhance the sharing of these standalone model instances. Also, this work  
110 could later be merged with past work on model component metadata to create an overarching  
111 model metadata framework.

112       A motivating factor for this research is the design and development of a new system  
113 called HydroShare (<https://www.hydroshare.org>). The goal of HydroShare is to advance  
114 hydrologic science by enabling the scientific community to more easily and freely share products

115 resulting from their research – not just the scientific publication summarizing a study, but also  
116 the data and models used to create the scientific publication (Horsburgh et al., 2015; Tarboton et  
117 al., 2014; Tarboton et al., 2013). HydroShare is a web-based collaborative system developed  
118 with the goal of sharing, accessing, and discovering hydrologic data and models (Tarboton et al.,  
119 2013). It was designed and built by the authors, along with a larger team of researchers, in  
120 collaboration with the Consortium of Universities for the Advancement of Hydrologic Science,  
121 Inc. (CUAHSI).

122         The basic unit of digital content in HydroShare is called a “resource.” One of the key  
123 steps in designing HydroShare was defining metadata for different resource types (Horsburgh et  
124 al., 2015; Tarboton et al., 2014). While users can upload any digital content as a “generic  
125 resource” within HydroShare, these generic resources only support basic metadata elements  
126 defined by the Dublin Core metadata framework that are applicable to any data type. Specific  
127 resource types in HydroShare can extend this Dublin Core metadata to provide new metadata  
128 elements that support functionality specific to common hydrologic datasets (Horsburgh et al.,  
129 2015). For example, the time series resource types support additional metadata terms relevant to  
130 a time series, and the system can automatically plot time series resources because of this  
131 metadata (Sadler et al., 2015). Because a model metadata framework like this did not exist for  
132 hydrologic models, we first had to design one. Then, we used the model metadata framework we  
133 designed in HydroShare to implement new resource types specific to the needs of hydrologic  
134 models. While the HydroShare implementation motivated the design of the model metadata  
135 framework, it is important to emphasize that the metadata framework described here is general  
136 and can be adopted across cyberinfrastructure systems.

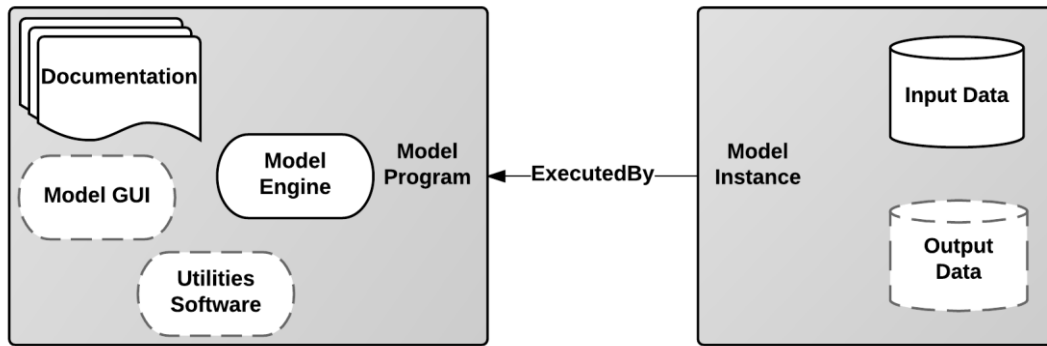
137           The remainder of the paper is organized as follows. First, a Methodology section is  
138 presented discussing the design of the model metadata framework and describing an example use  
139 case where the design implemented in HydroShare was used to share results from a hydrologic  
140 modeling study. Next, the Results section presents the implemented software and the results  
141 from the example use case. Finally, the paper concludes with a summary discussion of the  
142 proposed approach and steps that could be taken to further advance this work.

143

## 144 **2. Methodology**

### 145 *2.1. Metadata Framework Design*

146           The metadata framework design considers a computational model as two distinct  
147 concepts: 1) a *model program resource*, which includes software for executing a model  
148 simulation and generating outputs, and 2) a *model instance resource*, which includes the input  
149 files and, optionally, the output files for a specific simulation. The Resource Description  
150 Framework (RDF) is used for defining concepts and their associated metadata using a subject,  
151 predicate, and object structure (<http://www.w3.org/RDF>). As a simple example, this basic  
152 structure can be used as illustrated in Figure 1 to show that a model instance (subject) is executed  
153 by (predicate) a model program (object). Each resource has core metadata defined by the Dublin  
154 Core metadata framework and extended metadata designed through this research. Details of the  
155 metadata for model programs and model instances are described in the following subsections.



156

157 Figure 1. Key components of the model program and model instance resources.

158 2.1.1. Model Program Resource Metadata

159 The model program resource encapsulates all of the software and files necessary to  
 160 identify, install, and run a given hydrologic model. The model program includes a model engine,  
 161 which is the core mathematical modeling logic for the model (Morsy et al., 2014). This model  
 162 engine is often, but not always, embedded within a larger application that includes visualization,  
 163 typically using a graphical user interface (GUI), and other utility software. It is not uncommon  
 164 for multiple model programs to use the same or similar model engine; for example, there are  
 165 multiple model programs with different user interfaces that all use the Storm Water Management  
 166 Model (SWMM) as its model engine. A key design decision was to link a model program with a  
 167 model instance, rather than a model engine with a model instance. This was done because  
 168 developers may make subtle but important changes to publically available model engines within  
 169 their own model programs. Thus, it is difficult to guarantee that two independent model  
 170 programs, both making use of the same original model engine, will produce the exact same  
 171 output.

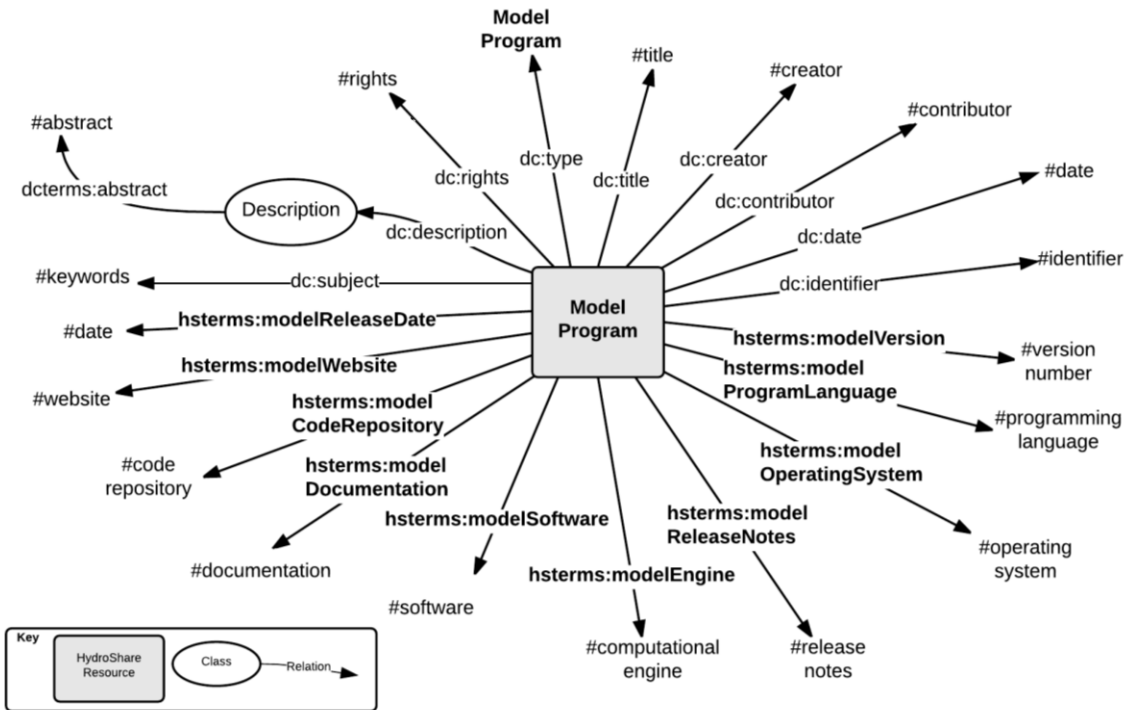
172 The goal when identifying metadata for a model program was to sufficiently describe a  
 173 specific version of the software, its computer system compatibility, as well as its proper and



174 intended use. To foster interoperability, this metadata consists of a basic description of the  
175 resource using the Dublin Core metadata standard (shown using the “dc” and “dcterms” prefixes)  
176 that is then extended with resource specific metadata (Figure 2; Table 1). These extended  
177 metadata terms are given the “hsterms” prefix, indicating that they belong to a namespace of  
178 terms defined by HydroShare, and are subdivided into content-related and resource-related  
179 categories. Content-related metadata includes items such as the *computational engine*, *software*,  
180 *release notes*, and *documentation* to describe the content that should accompany a model  
181 program resource. A model program is required to include a model engine, while the other  
182 content-related metadata items are optional.

183         The resource-related metadata describe characteristics of a model program using high-  
184 level terminology with the aim of clearly defining and distinguishing between similar model  
185 program resources. These include *release date*, *website*, *version*, *language*, *software repository*,  
186 and *operating system* metadata. The release date element provides general information about the  
187 hydrologic model to aid in version identification, while the website element is intended to  
188 provide users additional model-specific information. The remaining elements describe the  
189 software attributes and system compatibility of the model program as shown in Table 1. These  
190 metadata terms can serve many different uses, including enhanced search and discovery across a  
191 large collection of model program resources. They also aim to support reproducibility by  
192 capturing the exact model program used to execute a particular model instance.

193



194

195 Figure 2. Model program resource metadata elements as RDF triples. The # prefix signifies an  
 196 attribute that can be populated when implementing the metadata framework for a given model  
 197 program.

198 Table 1. Model program extended metadata element definitions.

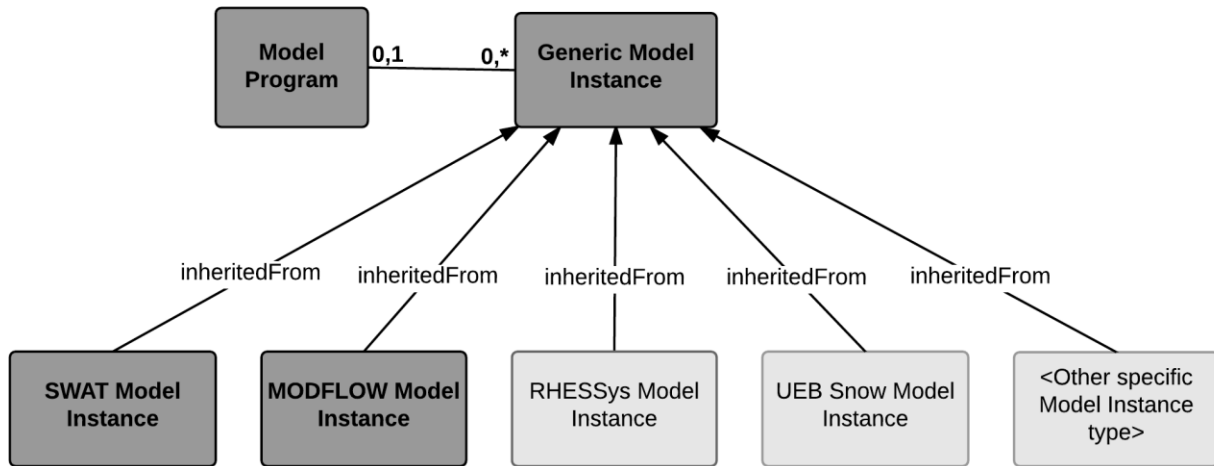
Metadata Term	Cardinality	Definition
hsterms:modelVersion	1..1	Unique model version and/or build number
hsterms:modelProgramLanguage	0..*	The programming language(s) used to write the model program
hsterms:modelOperatingSystem	0..*	Compatible operating system(s) for executing the model program
hsterms:modelReleaseDate	0..1	The date that this version was released
hsterms:modelReleaseNotes	0..*	Notes regarding the release
hsterms:modelWebsite	0..1	A URL to the website maintained by the model developers
hsterms:modelCodeRepository	0..*	A URL to the source code repository (e.g., Github, Bitbucket, etc.)
hsterms:modelDocumentation	0..*	Documentation related to the model (e.g., User manual, theoretical manual, reports, etc.)
hsterms:modelSoftware	0..*	The model program software (e.g., source code, installer, utilities, etc.)
hsterms:modelEngine	0..*	The model engine (e.g., source code, binary, executable, etc.)

199

200 2.1.2. Model Instance Resource Metadata

201 The model instance resource describes the input files required for execution by a model  
 202 program. A model instance resource may optionally include the output files resulting after  
 203 execution. The design for metadata associated with a model instance was intended to capture the  
 204 aspects required to define and distinguish between different model instances across the wide  
 205 variety of hydrologic models. To accomplish this, the design first includes a *generic model*  
 206 *instance*. This generic model instance has metadata elements applicable to any model program.  
 207 The design also includes *specific model instances* that inherit the properties of a generic model  
 208 instance and add new properties that are relevant to one or more model programs. This pattern is  
 209 illustrated in Figure 3. In this figure, some specific model instance resources are listed as  
 210 examples, with the idea that this list can be extended to include other hydrologic models as well.

211 This design, therefore, provides two ways to capture metadata for a model instance. The default  
 212 option would be to use a generic model instance resource type. However, if available, a specific  
 213 model instance resource type tailored for the model program used to execute that model instance  
 214 should be used instead for enhanced functionality and metadata capture.



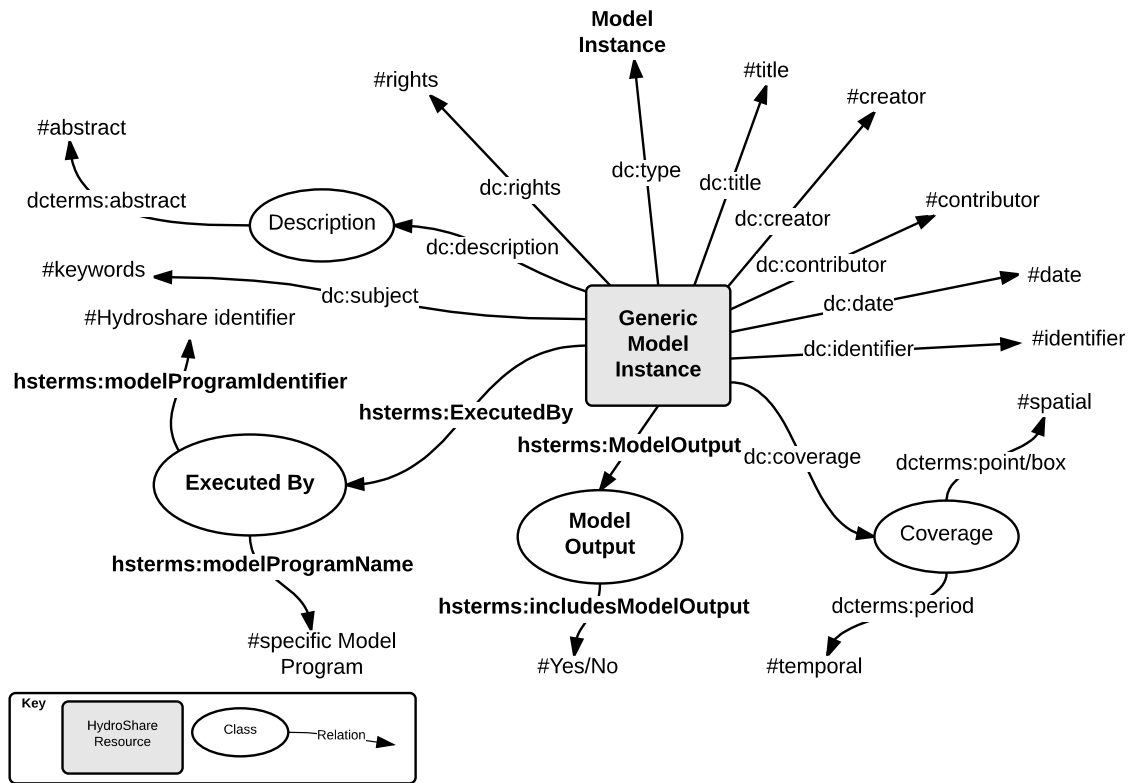
215  
 216 Figure 3. Generic model instance and specific model instance hierarchy. Model program, generic  
 217 model instance, SWAT model instance, and MODFLOW model instance metadata have already  
 218 been designed, while metadata for the other specific model instances are either in development  
 219 or planned for the near future.

220 Figure 4 presents the metadata for a generic model instance. Because the generic model  
 221 instance extends the Dublin Core metadata framework, it inherits the metadata terms defined by  
 222 Dublin Core (shown using the “dc” and “dcterms” prefix). One metadata element defined in  
 223 Dublin Core that is particularly important for model instances is the coverage element. This  
 224 metadata element defines the temporal and spatial extent of a resource. For a model instance  
 225 resource, the temporal coverage provides the start and end date/time for the simulation; the  
 226 spatial coverage provides a place name and geographic coordinates for the model instance. The

227 spatial coverage can be represented by a point (e.g., the centroid of the modeling domain) or a  
228 box (e.g., the bounding box of the modeling domain). This coverage element does not represent  
229 the exact shape of the model instance, but rather its geographic location or extent.

230         The generic model instance is extended with additional metadata elements having the  
231 “hsterms” prefix (Figure 4; Table 2). These metadata elements are subdivided into two main  
232 classes: ModelOutput and ExecutedBy. ModelOutput includes information about the output data  
233 generated by the model after it is executed. Only one element was deemed necessary in the initial  
234 design for describing the model output, although more elements could be added later. The  
235 element included is includesModelOutput, which allows users to indicate if the output files are  
236 included along with the input files as part of the model instance resource. The ExecutedBy  
237 element links the model instance resource with the model program resource that is used for  
238 execution. ExecutedBy includes two sub-metadata elements: modelProgramName and  
239 modelProgramIdentifier. The modelProgramName element stores the name of the linked model  
240 program resource, while modelProgramIdentifier stores its unique identifier. By linking a model  
241 instance to a model program resource, the ExecutedBy metadata element facilitates later  
242 reproducibility of the model results.

243



244

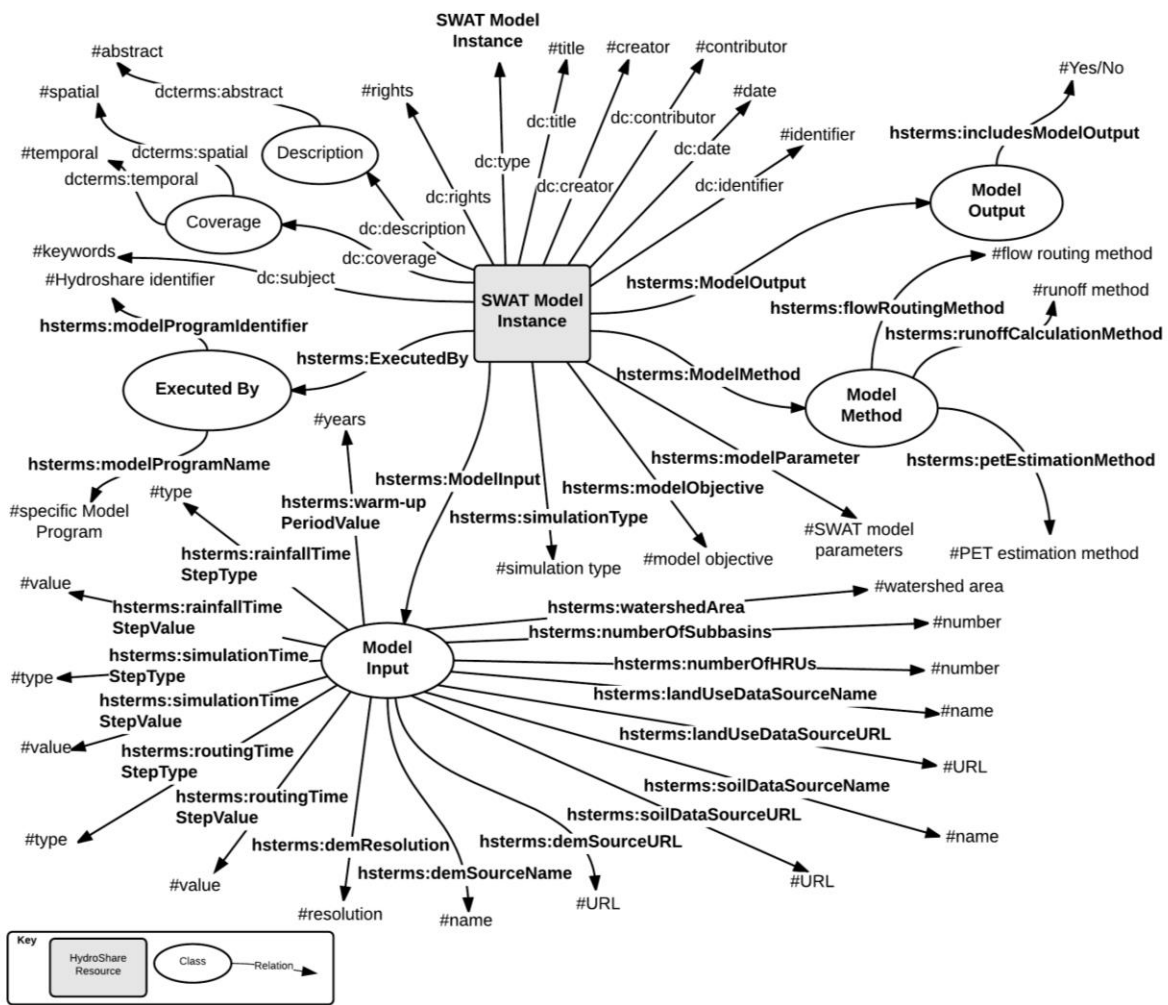
245 Figure 4. Generic model instance resource metadata elements as RDF triples.

246 Table 2. Generic model instance extended metadata element definitions.

Metadata Term	Cardinality	Definition
<b>hsterms:modelOutput</b>		A class used for describing output for an executed model instance
hsterms:includesModelOutput	1..1	A boolean value that indicates if the output files are included with the model instance
<b>hsterms:executedBy</b>		A class that describes the model program that executes the model instance
hsterms:modelProgramName	0..1	The name of the model program that executes the model instance
hsterms:modelProgramIdentifier	0..1	The identifier for the model program that executes the model instance

247

248 As an example of a specific model instance, consider an extension to the generic model  
 249 instance designed to add metadata specific to an instance of the Soil and Water Assessment Tool  
 250 (SWAT). This SWAT model instance offers extended metadata elements that more fully  
 251 describe SWAT model instances, but that are not directly applicable to other hydrologic models.  
 252 It was designed to be compatible with the SWATShare application, which is an interactive Web  
 253 tool used to run, visualize, and interact with the SWAT model instances (Rajib et al., 2016). The  
 254 extended metadata elements for a SWAT model instance are shown in Figure 5, and the extended  
 255 metadata terms are defined in Table 3.



256  
 257 Figure 5. SWAT model instance metadata as RDF triples.

258 Table 2. SWAT model instance extended metadata element definitions.

Metadata Term	Cardinality	Definition
hsterms:modelObjective	1..*	The objective of the model (e.g., hydrology, water quality, BMPs, climate / landuse change, etc.)
hsterms:simulationType	0..1	The type of the simulation used (e.g., normal simulation, sensitivity analysis, and auto-calibration)
<i>hsterms:modelInput</i>		Class for describing the model instance input files
hsterms:warm-upPeriodType	0..1	The warm-up period type (always years)
hsterms:warm-upPeriodValue	0..1	The numeric value of the warm-up period in years
hsterms:rainfallTimeStepType	0..1	The type of time step used in the simulation for input rainfall data (e.g., daily or sub-hourly)
hsterms:rainfallTimeStepValue	0..1	The time step value associated with the rainfall data
hsterms:routingTimeStepType	0..1	The type of time step used in the simulation for river routing calculations (e.g., daily or hourly)
hsterms:routingTimeStepValue	0..1	The time step value used for the river routing calculations
hsterms:simulationTimeStepType	0..1	The type of time step type used for model simulation (e.g., annual, monthly, daily, or hourly)
hsterms:simulationTimeStepValue	0..1	The time step value used for simulation
hsterms:watershedArea	0..1	The watershed area in km <sup>2</sup>
hsterms:numberOfSubbasins	0..1	The number of subbasins within the watershed
hsterms:numberOfHRUs	0..1	The number of hydrologic response units (HRUs) within the watershed
hsterms:DEMResolution	0..1	The resolution of the digital elevation model (DEM) in meters
hsterms:DEMSourceName	0..1	The name of the DEM provider
hsterms:DEMSourceURL	0..1	The URL of the DEM
hsterms:landUseDataSourceName	0..1	The name for the land use / land cover (LULC) dataset provider
hsterms:landUseDataSourceURL	0..1	The URL for the LULC dataset
hsterms:soilDataSourceName	0..1	The name for soil dataset provider
hsterms:soilDataSourceURL	0..1	The URL for Soil dataset
<i>hsterms:modelMethod</i>		Class that describes the model methods used in the simulation
hsterms:runoffCalculationMethod	0..1	The runoff calculation method used
hsterms:flowRoutingMethod	0..1	The flow routing method used
hsterms:PETEstimationMethod	0..1	The Potential EvapoTranspiration (PET) estimation method used
hsterms:modelParameter	0..*	The parameters used in the model (e.g., crop rotation, title drainage, point source, fertilizer, tillage operation, inlet of draining watershed, irrigation operation, etc.)

259

260 While SWAT is used to provide an example of a specific model instance, similar  
 261 metadata could be developed for other models. The design goal of this work, however, is not for  
 262 our team to capture metadata relevant to all hydrologic models, as doing so would be  
 263 impractical. Rather, our goal was to design a framework that has a common core and a clear



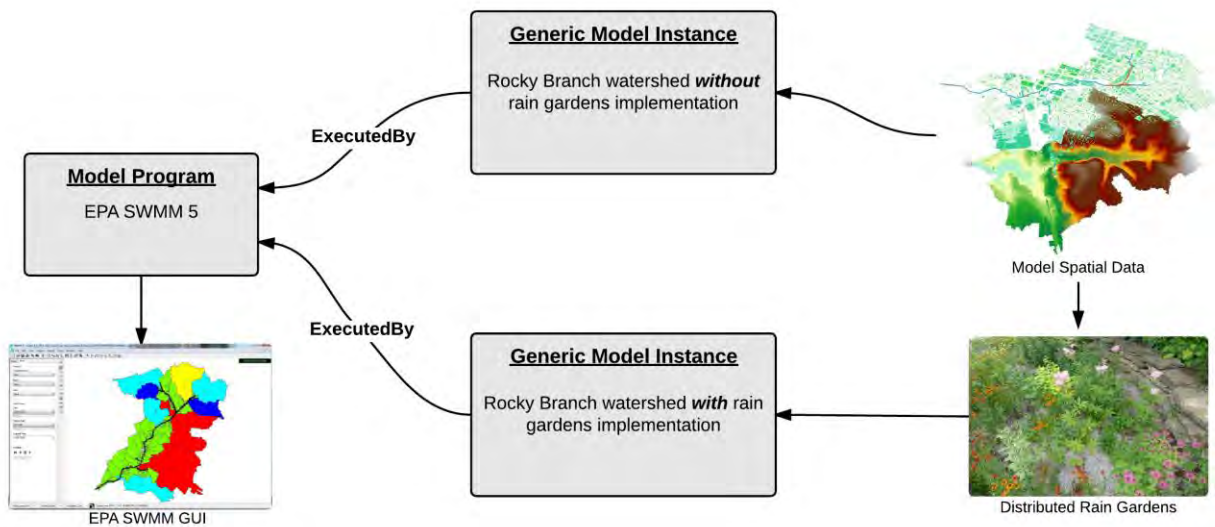
264 methodology for extending this core for specific hydrologic models. We plan to provide  
265 examples, like the SWAT example, that third party developers can follow to create their own  
266 specific model instance metadata. By providing a common foundation for metadata and  
267 resource-structure across models, there will be a level of standardization that will aid in  
268 interoperability across software systems. Specific model metadata acknowledges the diversity  
269 among hydrologic models and does not force conformity to a single set of metadata elements.  
270 The design also allows for changes in the future. For example, if additional common model  
271 metadata elements are identified across hydrologic models, then they can be added to the generic  
272 model instance class and inherited by all specific model instances.

273

## 274 2.2. *Experimental Use Case*

275 To demonstrate the metadata design, we used the application of SWMM to study flooding in  
276 an urban watershed, from prior research (Morsy et al., 2016), as a use case. We now wish to  
277 publish the resulting model instances online. There are many motivating factors for doing this.  
278 First, we believe that a model instance, like the journal paper, is an important product from the  
279 research and should stand on its own as a citable product. Second, we want to foster ways for  
280 other scientists to build from or reuse our model to address their own scientific research  
281 questions. Third, we want to ensure that the model program used in our study, including the  
282 model engine, utility software, and documentation, is captured within a single online resource.  
283 This is important because, after some time, the model program developers may not provide this  
284 particular version of the software on their website. Lastly, this is a way of meeting the sponsor's  
285 data management obligations for the research.

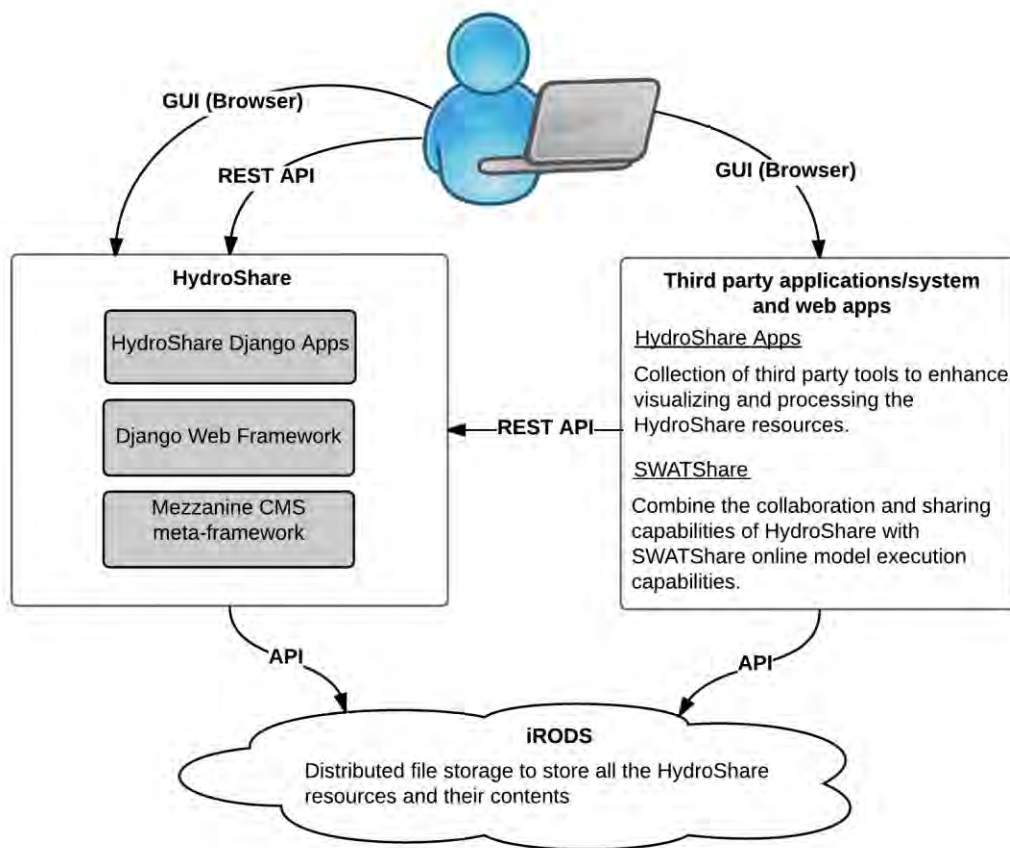
286 The objective of this prior modeling study was to better understand the potential of rain  
 287 gardens as distributed stormwater controls for flood mitigation within an urbanized watershed  
 288 (Morsy et al., 2016). The specific study area of the research was the Rocky Branch watershed,  
 289 which is located in downtown Columbia, South Carolina, USA. Because a significant portion of  
 290 the watershed is developed, high intensity storms that typically occur during the summertime  
 291 result in flooding at different locations within the watershed. For this study, two different model  
 292 instances were created (Figure 6). The first model instance is a well-calibrated and evaluated  
 293 model that simulates flooding events in the Rocky Branch watershed. The second model instance  
 294 builds from the first model instance and includes additional, hypothetical rain gardens as  
 295 stormwater controls to test if their addition mitigates flooding in the watershed for storm events  
 296 with different return periods.



297  
 298 Figure 6. Use case implementation as a model program and two model instance resource types.

299 The metadata framework was implemented within HydroShare and used to share the  
 300 model program and model instance resources for the example application. HydroShare, as  
 301 introduced earlier, is an online system for managing resources adhering to a Resource Data

302 Model (Horsburgh et al., 2015; Tarboton et al., 2013; 2014). The HydroShare architecture  
 303 consists of open source components including Django, a web application platform, Mezzanine, a  
 304 content management system meta-framework, and the Integrated Rule-Oriented Data System  
 305 (iRODS), an enterprise storage management middleware (Rajasekar et al., 2010) organized as  
 306 shown in Figure 7 (Heard et al., 2014). Results detailing the technical aspects of the software  
 307 implementation are presented in Section 3.1.



308

309 Figure 7. HydroShare's general architecture emphasizing the connections between the user,  
 310 HydroShare, iRODS, and third party applications

311 Although a SWMM-specific model instance resource type could have been designed and  
 312 implemented within HydroShare, we used the generic model instance resource type when

313 implementing the use case to provide an example applicable to any hydrologic model. A  
314 SWMM-specific model instance would have allowed for the capture of additional metadata  
315 relevant only to SWMM models. Software extensions to HydroShare could then provide custom  
316 functionality and applications able to operate specifically on SWMM-model instances. Using the  
317 generic model instance offers broad use across hydrologic models, but it lacks the potential for  
318 customization that becomes possible when targeting a specific model instance resource type.

319

### 320 **3. Results**

#### 321 *3.1. Results for Software Implementation within HydroShare*

322 Figure 8 shows the class structure for the new model resource types implemented within  
323 HydroShare based on the metadata framework design. Each resource type consists of three main  
324 categories of classes: the resource data type class, the classes for the individual extended  
325 metadata elements, and the container class that groups all metadata elements. For example, the  
326 classes in the three categories for the model instance resource type are 1)  
327 ModelInstanceResource, which is the resource data type class, 2) ModelOutput and ExecutedBy,  
328 which are the classes representing the extended metadata elements, and 3)  
329 ModelInstanceMetaData, which is the class that contains all the metadata elements. The resource  
330 type classes for model instance and model program inherit from the BaseResource class, which,  
331 in turn, inherits from the Abstract Resource class. This structure allows the model resource type  
332 to inherit the Dublin Core metadata elements. Specific model instance metadata, like that for the  
333 SWAT model instance resource type, inherits from the generic model instance resource type  
334 class. The diagram shown in Figure 8, therefore, could be extended for other specific model  
335 instance metadata.



345 class, which is the metadata container that includes the common metadata element objects. These  
346 classes are the link between the ModelProgramResource, the ModelInstanceResource classes,  
347 and their extended metadata classes. One-to-one relationships are made between  
348 ModelProgramMetaData and ModelInstanceMetaData classes and each of their respective  
349 extended metadata classes. These extended metadata classes are then included as supported  
350 metadata elements for their related resources (ModelProgram or ModelInstance resources) where  
351 they could be used to create, update, and delete class instances associated with these resource  
352 types.

353 An important method of the CoreMetaData, ModelProgramMetaData,  
354 ModelInstanceMetaData, and SWATModelInstanceMetaData is `get_xml`. This method converts  
355 the stored metadata into an RDF-XML format. The `CoreMetaData.get_xml` method extracts the  
356 generic metadata elements, while the `get_xml` method for each specific resource extracts the  
357 related extended metadata elements. For example, for a ModelInstance resource, the  
358 `CoreMetaData.get_xml` method is used to extract the Dublin Core standard metadata elements,  
359 while the `ModelInstanceMetaData.get_xml` method is used to extract the extended metadata  
360 elements.

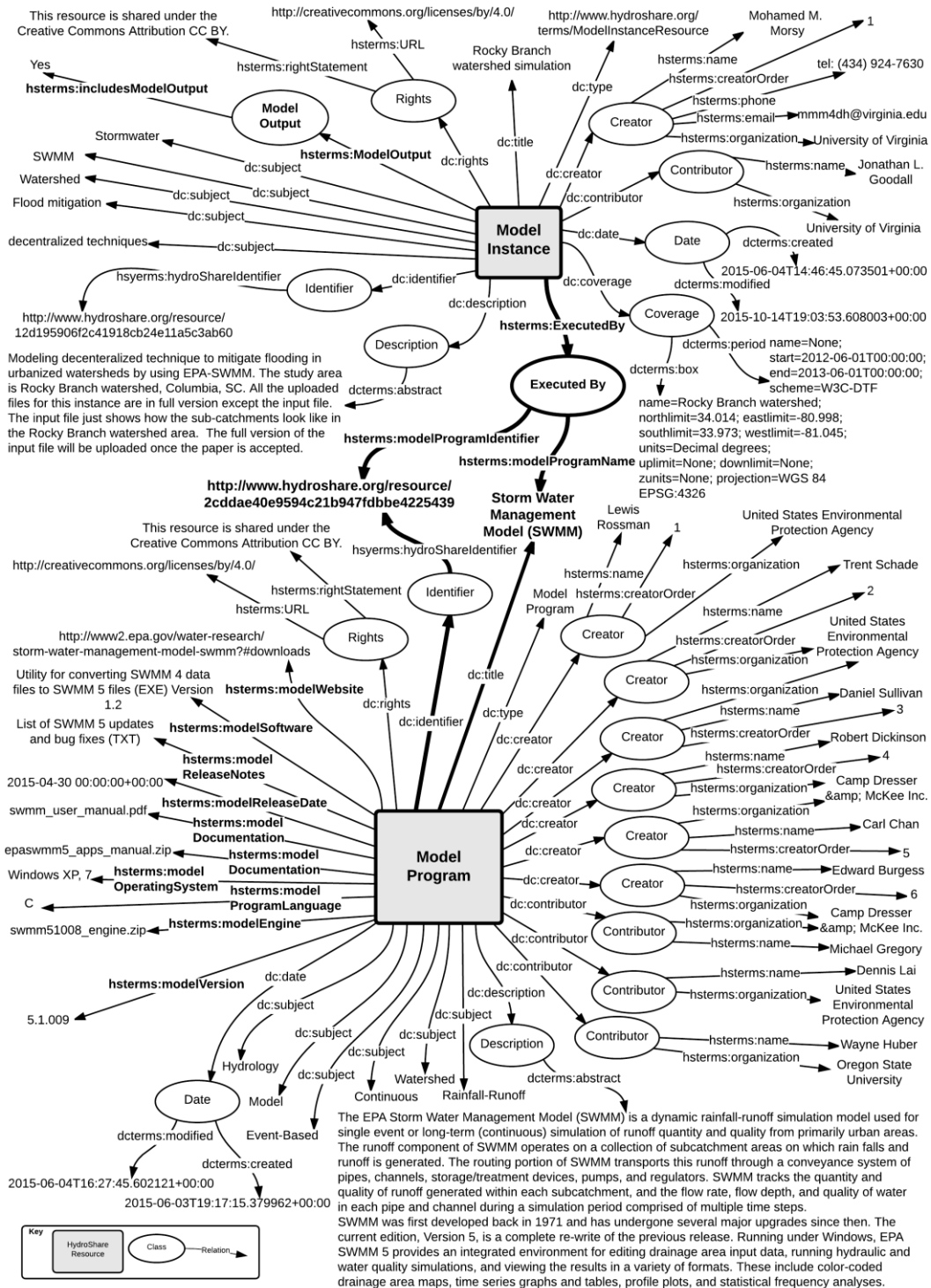
361

### 362 *3.2. Results from the Example Use Case*

363 Figure 9 illustrates the metadata that can be captured for the example use case using the  
364 generic model instance and model program resources. Each resource has a title, creator, and  
365 other metadata that follow the Dublin Core metadata standard. In addition, extended metadata for  
366 each resource (shown using the “hsterms” prefix) help to more fully describe the model instance  
367 and corresponding model program used for executing the model instance. Figure 9 also shows

368 how the model program resource type, in this case the SWMM model (Rossman et al., 2016),  
369 and the model instance resource type, in this case a Rocky Branch watershed simulation, are  
370 connected using the ExecutedBy relationship.

371         Figure 10 is an activity diagram showing the steps used to create new model resources on  
372 hydroshare.org. Three resources were created in this example: a model program resource for the  
373 EPA-SWMM model version 5.1.009 (Rossman et al., 2016) and two model instance resources  
374 for the Rocky Branch watershed simulations (e.g., Morsy, 2015). Figure 11 shows the Graphical  
375 User Interface (GUI) for how a user selects a model resource type within HydroShare. In the  
376 current implementation, the model resource types are grouped together under the modeling title.  
377 Once the user selects the desired resource type, adds a title, and uploads the related files, the new  
378 resource is created in HydroShare and the user sees the landing page for this newly created  
379 resource. At this point, a unique identifier specific to the HydroShare system has been  
380 automatically assigned to the resource. Later, if the user decides to formally publish the resource  
381 in HydroShare, a more formal digital object identifier (DOI) would be assigned to the resource.  
382 After a resource is formally published and a DOI is assigned, the user is no longer allowed to  
383 make changes to the resource metadata or the uploaded files. Prior to formal publication,  
384 authorized users can make changes to the resource at any time.

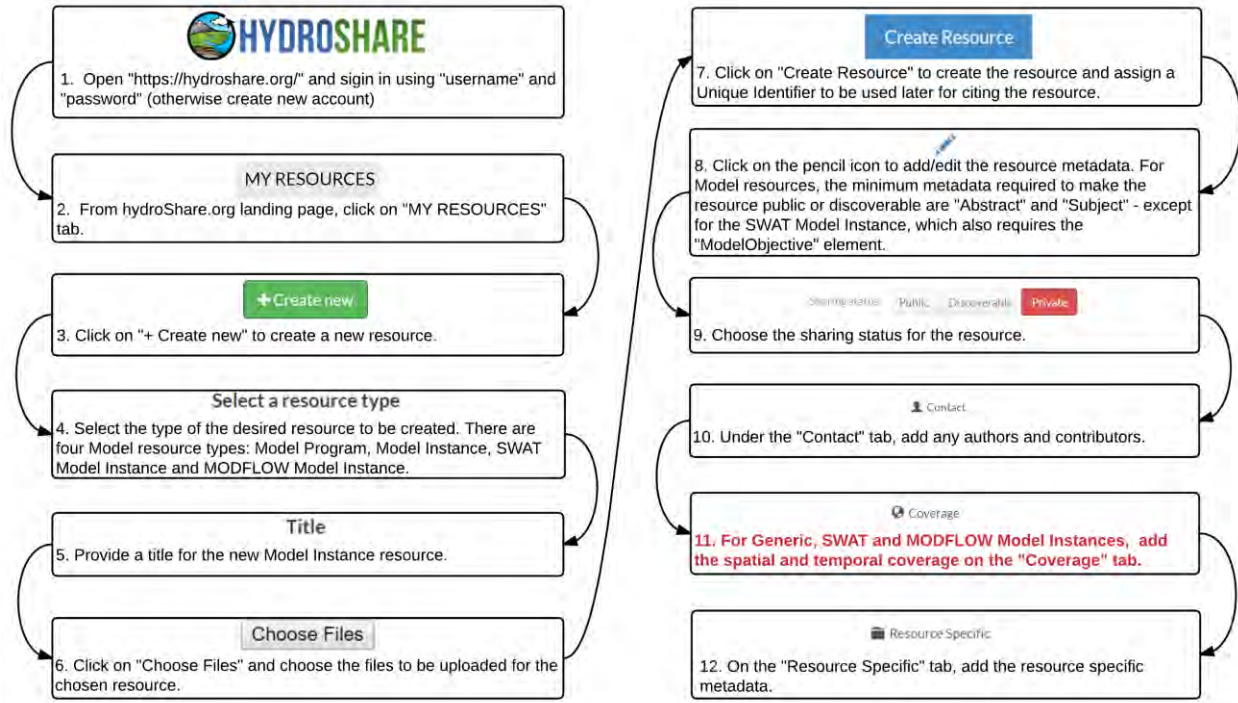


385

386 Figure 9. Results of populating the model instance and model program metadata for the example

387 use case.





389

390 Figure 10. Activity diagram to describe the steps required to create new model resource type  
 391 within HydroShare. Step 11 is highlighted to indicate that only model instances require coverage  
 392 and not model programs.



## Select a resource type and upload files as needed to create a new resource

- Files you upload here will be grouped together into a "Resource" in HydroShare.
- File size is limited to 1 GB per file for direct browser upload from your local disk.
- Files larger than 1GB can be imported directly from any IRODS account. You can create a HydroShare IRODS account from your [user profile](#) page. See help for information on working with IRODS directly.
- You can also add files directly from any IRODS server regardless of the file size.

Select a resource type

Title

Add your files here:

Any file type can be uploaded.  
Multiple file upload is allowed.

To load files from your local disk

No file chosen

To load files from an IRODS server

[log in to IRODS](#)

Generic

**Generic**

Multidimensional (NetCDF)

Web App

Collection Resource

**Geographic Resources**

Geographic Raster

Geographic Feature (ESRI Shapefiles)

**Time Series Resources**

Time Series

HIS Referenced Time Series

**Modeling**

Model Program

Model Instance

SWAT Model Instance

MODFLOW Model Instance

Script

⇒ Model Resources in Hydroshare

393

394 Figure 11. Screen shot showing model resource types currently implemented on hydroshare.org.

395           Figures 12 and 13 show the resource specific metadata for the model program resource

396 and the generic model instance resource types, respectively, on their landing pages in

397 HydroShare. These figures show HydroShare’s metadata “edit” mode to illustrate all of the

398 available metadata elements, as HydroShare’s default is to hide metadata elements for which

399 there are no values in regular “view” mode. Note that the model instance is linked to the model

400 program used for execution (Figure 13). Under the “Model Program used for execution” heading

401 on the generic model instance landing page, there is a dropdown list that collects all the available

402 public model program resources in HydroShare. The user chooses the model program resource

403 used to execute the model instance resource from the dropdown list (or adds a new model

404 program resource if it is not already available). Once the user chooses the desired model program

405 resource, a summary of the model program metadata is displayed to aid the user in confirming

406 that the correct model program was selected.

407 Another important aspect of the model instance resource is the coverage metadata. Figure  
408 14 shows how the coverage metadata appears in the Coverage tab on the resource's landing page.  
409 As explained above, there are two types of coverage metadata elements: spatial and temporal. All  
410 of the spatial metadata is expressed in World Geodetic System (WGS) 84 coordinates, which is  
411 used throughout HydroShare. For the use case, the spatial metadata was entered for this model  
412 instance as a two-dimensional bounding box (rather than an XY point). Once the user inserts the  
413 bounding coordinates, the box will appear on the map so that the user can confirm the spatial  
414 coverage extent. The user can also specify the coverage by clicking a point on the map or  
415 dragging a box on the map. The temporal coverage metadata consist of start and end dates for the  
416 model instance. HydroShare uses this coverage metadata to support both spatial (e.g., map-  
417 based) and temporal searches to identify relevant resources.

HYDROSHARE MY RESOURCES DISCOVER APPS HELP

Contact Coverage Related Resources Resource Specific

### Data

**Computational Engine**  
 swmm51008\_engine.zip

**Software**  
 swmm.exe

**Documentation**  
 2 selected

- swmm.exe
- swmm\_user\_manual.pdf
- swmm51008\_engine.zip
- epaswmm5\_apps\_manual.zip

### General

**Release Date**  
 04/30/2015

**Version**  
 5.1.009

### Software

**Website**  
 http://www2.epa.gov/water-research/storm-water-management-model-swmm?download

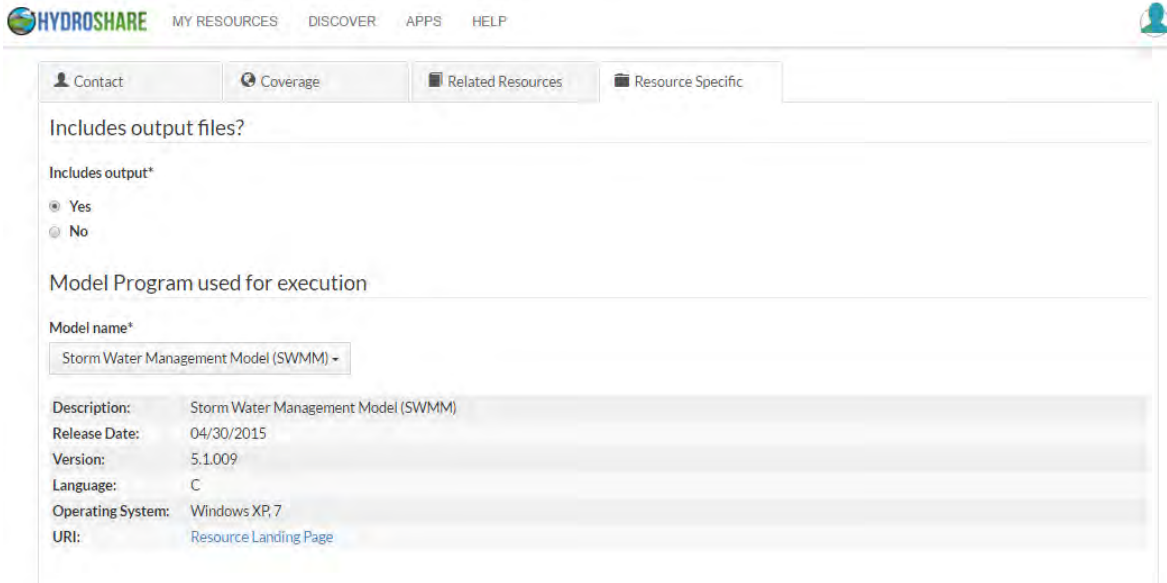
**Language**  
 C

**Operating System**  
 Windows XP, 7

**Software Repository**

418

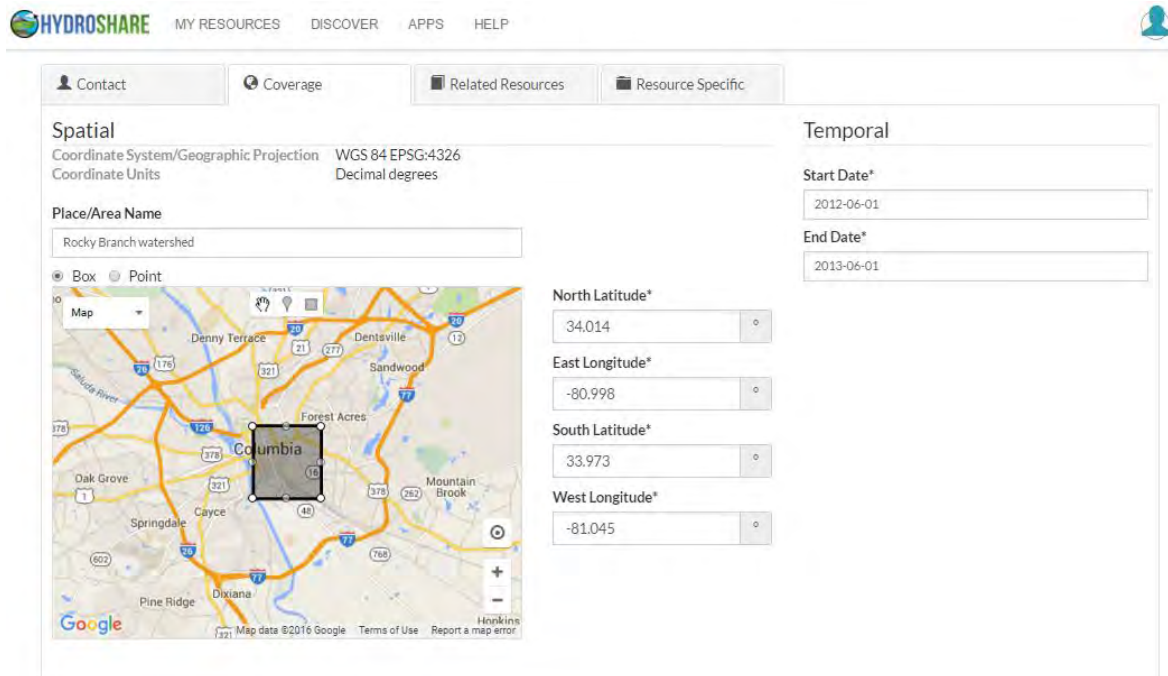
419 Figure 12. Model program resource specific metadata on the resource’s landing page on  
 420 hydroshare.org (shown in edit mode).



421

422 Figure 13. Generic model instance resource specific metadata on the resource's landing page on  
423 hydroshare.org (shown in edit mode).

424



425

426 Figure 14. Model instance resource type coverage metadata on the resource’s landing page  
 427 (shown in edit mode) on hydroshare.org.

428 **4. Discussion**

429 One of the most difficult design decisions in this work was to separate model programs  
 430 and model instances into two different resources rather than a single combined resource. The  
 431 design decision was made for the following reasons. First, it allows the model program metadata  
 432 to be entered once within the system. Second, it simplifies the task of identifying all instances of  
 433 a given model program stored within the system. Third, it provides a path for online execution of  
 434 many model instances that are linked to a single model program. We felt these benefits  
 435 outweighed the added complexity and management needs introduced by separating the model  
 436 program and model instance concepts into different resources types. We acknowledge that some  
 437 use cases require incremental changes to a model program’s source code, and we are considering  
 438 options for capturing these incremental changes to model programs without the need to create a

439 completely new resource every time a model program's source code has been changed. That  
440 said, users are not restricted from uploading a model program within a model instance, if desired.  
441 If this becomes common practice, we have considered allowing a model instance resource's  
442 ExecutedBy field to point to itself. This would signify to a user that the model program, whether  
443 it be a compiled binary file or the source code, is located within the model instance resource.

444 Another key design decision was to allow a model instance resource to be linked to only  
445 one model program resource. We realize that it is possible for a model instance to be executed  
446 successfully by multiple model program resources (e.g., two model programs with different  
447 versions but compatible with the same model instance). However, allowing a model instance to  
448 be linked to more than one model program would introduce uncertainty about what program was  
449 used to execute the instance for a given study. Reproducibility could be compromised as a result,  
450 because executing the model instance with a different model program may return slightly  
451 different results. For this reason, the design requires a model instance to be linked to only one  
452 model program.

453 We encountered through the use case application the important issue of how to handle the  
454 case where the person uploading a resource into HydroShare, what HydroShare refers to as the  
455 resource's owner, is not the author of that resource. HydroShare separates intellectual credit  
456 attribution from access control and management of content. The Dublin Core vocabulary term  
457 "Creator" is used in HydroShare metadata for the intellectual originator of the content. This is  
458 displayed as Author on landing pages and used in citations. The term "Owner" is used in access  
459 control and management of content and is the HydroShare user typically responsible for  
460 uploading the content (although ownership can be transferred after uploading, and others can be  
461 assigned permissions to edit and upload content). In the SWMM model program resource

462 example, the EPA-SWMM model was authored by researchers at the United States  
463 Environmental Protection Agency (EPA) but, was uploaded to HydroShare by the modeler, one  
464 of the authors of this paper. The original authors of SWMM were entered as authors for the  
465 resource and the relationship “isCopiedFrom” was added to the resource pointing to the website  
466 from which the model program was obtained. With this added relationship, the HydroShare  
467 system automatically generates and displays a citation on the resource’s landing page that shows  
468 that the resource in HydroShare was replicated from an external source, as shown below. The  
469 user that uploaded the resource into HydroShare, but did not author the resource, remains the  
470 resource owner but rightly does not receive authorship credit for this resource within the citation.

471

472 Rossman, L., T.Schade, D.Sullivan, R.Dickinson, C.Chan, E.Burgess (2016). Storm Water Management Model  
473 (SWMM), Version 5.1.010 with Low Impact Development (LID) Controls, [http://www2.epa.gov/water-](http://www2.epa.gov/water-research/storm-water-management-model-swmm)  
474 [research/storm-water-management-model-swmm](http://www2.epa.gov/water-research/storm-water-management-model-swmm), accessed 4/4/2016, replicated in HydroShare  
475 at:<http://www.hydroshare.org/resource/2cddae40e9594c21b947fdbbe4225439>

476

477 A limitation of this work at its current stage is the ability to scale-up to support dozens of  
478 different specific model instance resource types. Ideally, the creation of new HydroShare  
479 resource types would be simple enough that it could be done by the broader community of model  
480 developers. Currently, however, the process of creating a new resource type within HydroShare  
481 is time consuming and requires advanced knowledge of the HydroShare system and architecture.  
482 One approach to address this would be to focus on simplifying the process for creating new  
483 resource types. Another possibility would be to alter the approach described in this paper so that  
484 specific model instances are not implemented as new resource types, but still can have extended  
485 metadata for specific model programs. In this case, all model instances would be uploaded using



486 a single resource type, but there would be a mechanism to filter the metadata fields available to  
487 the user once the user or system identifies the uploaded model instance as being a specific and  
488 known type (e.g., a SWAT model instance). More research is needed to test these alternative  
489 options in terms of their practicality, usability, and scalability within HydroShare.

490 A longer-term goal of this work is to provide server-side execution of model instances  
491 directly through HydroShare. By knowing and storing the exact model program used to execute a  
492 model instance within HydroShare, it should be possible to install the model program onto  
493 server-side computational resources and execute a model instance using these resources. The  
494 updated model instance including the newly generated output files could be automatically added  
495 to HydroShare via HydroShare's existing web service application programming interface (API),  
496 updating the original resource. Research on methods for achieving this goal, given the  
497 complexities of server-side model execution including the potential for large model instance  
498 sizes and long model execution times, has begun. Being able to execute a model instance directly  
499 through HydroShare could offer significant benefits including model reproducibility where a  
500 model run is performed in a controlled environment preconfigured with all required software  
501 dependencies.

502

## 503 **5. Conclusions**

504 This work presents a model metadata framework to support discovery, sharing and  
505 interpretation of hydrologic models. Key features of the framework are (1) that the model  
506 program and model instance are separate concepts with a one-to-many relationship (many  
507 instances may exist for a single model program), (2) that metadata for these concepts extend the  
508 well recognized and commonly used Dublin core metadata, and (3) that the model instance

509 concept is a hierarchy with a generic parent class implementable for any model program, and a  
510 more specific level tailored for certain model programs.

511 A key challenge in this or any other metadata framework design is providing the right  
512 balance between rich metadata for adequately describing details of resources and minimal  
513 metadata that is critical and can be easily populated. The growing number of generic data  
514 repositories available to hydrologists (e.g., figshare.com, zenodo.org, institutional repositories,  
515 etc.) largely adopt a minimal metadata approach. These systems provide metadata roughly  
516 equivalent to the metadata used to describe a generic resource in the HydroShare system. While  
517 this generic metadata could be used to describe, share, and discover model programs and model  
518 instances, it misses many of the important properties of these resources that could be leveraged  
519 for improved search, discovery, and use of model resources. The purpose of the metadata  
520 analysis and design presented here is to provide a more thorough, detailed metadata approach for  
521 model programs and instances. We expect to improve this metadata design over time as lessons  
522 are learned from its use, and as progress is made within the broader metadata and scientific  
523 modeling communities.

524 With the growing number of systems that serve a role within the larger  
525 cyberinfrastructure being built to support science, interoperability between these systems is  
526 becoming a more pressing need. If these systems are built from an agreed upon metadata  
527 framework, then it simplifies the transfer of resources between the systems. This would  
528 encourage each system to specialize in selected use cases while relying on external systems to  
529 handle other use cases outside of its scope. For example, in this work HydroShare specializes in  
530 model metadata, resource sharing, and resource publication. In ongoing research, we are building  
531 interoperability with the external SWATShare system that focuses on SWAT model execution

532 and visualization (Rajib et al., 2016). By adopting the same metadata and resource file structure  
533 for a SWAT model instance, these model instance resources can be more easily transferred  
534 between the two systems, and users can benefit from the functionality and strengths of both  
535 applications.

536 Future work will be aimed at improving the usability of the model program and model  
537 instance resources within HydroShare. For example, to reduce the time spent manually  
538 completing metadata fields, new functionality is planned to automate metadata extraction when a  
539 resource is uploaded and the metadata are already present within files uploaded with the  
540 resource. Model instances, for example, often include input files containing information on  
541 spatial and temporal coverage. The system should read these files, extract whatever metadata it  
542 can, and request only missing metadata fields from the user. This approach is difficult, however,  
543 given the diversity among hydrologic models; extracting metadata directly from model input  
544 files may require a significant amount of custom code. One potential long term benefit of this  
545 work would be for all model developers to add functionality that outputs a standard metadata file  
546 that can be read by HydroShare and other systems. Ideally, this would be done within the model  
547 program source code itself, but it could also be implemented as an external utility program.  
548 HydroShare and other systems could then read this file for automatic metadata extraction.

549

## 550 **6. Acknowledgements**

551 This work was supported by the National Science Foundation under collaborative grants ACI-  
552 1148453 and ACI-1148090. We acknowledge the work of the larger HydroShare development  
553 team.

554

555 **7. References**

- 556 Argent, R.M., 2004. An overview of model integration for environmental applications -  
557 Components, frameworks and semantics. *Environmental Modelling and Software*. 19, 219–  
558 234. doi:10.1016/S1364-8152(03)00150-6
- 559 Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore,  
560 R.W., 2016. Using a data grid to automate data preparation pipelines required for regional-  
561 scale hydrologic modeling. *Environmental Modelling and Software*. 78, 31–39.  
562 doi:10.1016/j.envsoft.2015.12.010
- 563 Boyko, A., J. Kunze, J. Littman, L. Madden, and B. Vargas, 2012. The BagIt File Packaging  
564 Format (v0.97), Network Working Group Internet Draft. [http://tools.ietf.org/html/draft-](http://tools.ietf.org/html/draft-kunze-bagit-10)  
565 [kunze-bagit-10](http://tools.ietf.org/html/draft-kunze-bagit-10), accessed August 2016.
- 566 David, C.H., Famiglietti, J.S., Yang, Z.-L., Habets, F., Maidment, D.R., 2016. A decade of  
567 RAPID-Reflections on the development of an open source geoscience code. *Earth and*  
568 *Space Science* . 3, 226–244. doi:10.1002/2015EA000142
- 569 Elag, M., Goodall, J.L., 2013. An ontology for component-based models of water resource  
570 systems. *Water Resoures Research*. 49, 5077–5091. doi:10.1002/wrcr.20401
- 571 Essawy, B.T., Goodall, J.L., Xu, H., Rajasekar, A., Myers, J.D., Kugler, T.A., Billah, M.M.,  
572 Whitton, M.C., Moore, R.W., 2016. Server-side workflow execution using data grid  
573 technology for reproducible analyses of data-intensive hydrologic systems. *Earth and Space*  
574 *Science*. 3, 163–175. doi:10.1002/2015EA000139
- 575 Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee,  
576 H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X.,  
577 2016. Towards the Geoscience Paper of the Future: Best Practices for Documenting and

578        Sharing Research from Data to Software to Provenance. Earth and Space Science  
579        doi:10.1002/2015EA000136

580 Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: Open modelling interface.  
581        Journal of Hydroinformatics 9, 175. doi:10.2166/hydro.2007.023

582 Heard, J., Tarboton, D., Idaszak, R., Horsburgh, J., Ames, D., Bedig, A., Castronova, A., Couch,  
583        A., 2014. An Architectural Overview Of HydroShare, A Next-Generation Hydrologic  
584        Information System. International Conference on Hydroinformatics. CUNY Academic  
585        Works. [http://academicworks.cuny.edu/cc\\_conf\\_hic/311](http://academicworks.cuny.edu/cc_conf_hic/311).

586 Hill, L., Crosier, S., Smith, T., Goodchild, M., 2001. A content standard for computational  
587        models. D-Lib Magazine 7, 6, 1082-9873.

588 Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M.J.,  
589        Tarboton, D.G., 2015. Hydroshare: Sharing Diverse Environmental Data Types and Models  
590        as Social Objects with Application to the Hydrology Domain. JAWRA Journal of the  
591        American Water Resources Association 52, 4. doi:10.1111/1752-1688.12363

592 ISO, 2003. ISO 19115:2003 Geographic Information -- Metadata.  
593        [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=26020](http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020), accessed August 2016.

594 ISO, 2011. ISO 19156:2011 Geographic Information -- Observations and measurements.  
595        [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=32574](http://www.iso.org/iso/catalogue_detail.htm?csnumber=32574), accessed August 2016.

596 Lagoze, C., Van de Sompel, H., Johnston, P., Nelson, M., Sanderson, R., Warner, S., 2008. Open  
597        Archives Initiative Object Reuse and Exchange: ORE User Guide – Primer  
598        <http://www.openarchives.org/ore/1.0/primer>, accessed August 2016.

599 Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G.,  
600        Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., 2013.

601 Integrated environmental modeling: A vision and roadmap for the future. *Environmental*  
602 *Modelling and Software* 39, 3–23. doi:10.1016/j.envsoft.2012.09.006

603 Michener, W., Vieglais, D., Vision, T., Kunze, J., Cruse, P., Janée, G., 2011. DataONE: Data  
604 Observation Network for Earth - Preserving Data and Enabling Innovation in the Biological  
605 and Environmental Sciences. *D-Lib Magazine* 17(1/2). DOI: 10.1045/january2011-  
606 michener.

607 Morsy, M.M., Goodall, J.L., Bandaragoda, C., Castronova, A.M., Greenberg, J., 2014. Metadata  
608 for Describing Water Models, in: *International Environmental Modelling and Software*  
609 *Society (iEMSs) 7th International Congress on Environmental Modelling and Software.*  
610 doi:10.13140/2.1.1314.6561

611 Morsy, M., 2015. Rocky Branch watershed simulation, HydroShare,  
612 <http://www.hydroshare.org/resource/12d195906f2c41918cb24e11a5c3ab60>.

613 Morsy, M.M., Goodall, J.L., Shatnawi, F.M., Meadows, M.E., 2016. Distributed Stormwater  
614 Controls for Flood Mitigation within Highly Urbanized Watersheds: Case Study for the  
615 Rocky Branch Watershed in Columbia, SC USA. *Journal of Hydrologic Engineering.*  
616 doi:10.1061/(ASCE)HE.1943-5584.0001430

617 Peckham, S.D., 2014. The CSDMS Standard Names: Cross-Domain Naming Conventions for  
618 Describing Process Models, Data Sets and Their Associated Variables. In: *Proceedings of*  
619 *the 7th International Congress on Environmental Modelling and Software*, D.P. Ames,  
620 N.W.T. Quinn, and A.E. Rizzoli (Editors). *International Environmental Modelling and*  
621 *Software Society (iEMSs)*, San Diego, California, ISBN: 978-88-9035-744-2.  
622 [https://csdms.colorado.edu/mediawiki/images/Peckham\\_2014\\_iEMSs.pdf](https://csdms.colorado.edu/mediawiki/images/Peckham_2014_iEMSs.pdf)

623 Peckham, S.D., Hutton, E.W.H., Norris, B., 2013. A component-based approach to integrated

624 modeling in the geosciences: The design of CSDMS. *Computers and Geosciences*. 53, 3–  
625 12. doi:10.1016/j.cageo.2012.04.002

626 Rajasekar, A., Moore, R., Hou, C.-Y., Lee, C. a., Marciano, R., de Torcy, A., Wan, M.,  
627 Schroeder, W., Chen, S.-Y., Gilbert, L., Tooby, P., Zhu, B., 2010. iRODS Primer:  
628 Integrated Rule-Oriented Data System, Synthesis Lectures on Information Concepts,  
629 Retrieval, and Services. doi:10.2200/S00233ED1V01Y200912ICR012

630 Rajib, Md. Adnan, Merwade, V. Luk Kim, I., Zhao, L., Song C. X., and Zhe, S. , 2016. A web  
631 platform for collaborative research and education through online sharing, simulation and  
632 visualization of SWAT models, *Environmental Modelling and Software*, 75, 498-512. doi:  
633 10.1016/j.envsoft.2015.10.032

634 Rossman, L., Schade, T., Sullivan, D., Dickinson, R., Chan, C., Burgess, E., 2016. Storm Water  
635 Management Model (SWMM), Version 5.1.010 with Low Impact Development (LID)  
636 Controls, <http://www2.epa.gov/water-research/storm-water-management-model-swmm>,  
637 accessed 6/2/2016, replicated in HydroShare at: <http://www.hydroshare.org/resource/2cdda>.

638 Sadler, J.M., Ames, D.P., Livingston, S.J., 2015. Extending HydroShare to enable hydrologic  
639 time series data as social media. *Journal of Hydroinformatics* jh2015331.  
640 doi:10.2166/hydro.2015.331

641 Singh, V.P., Frevert, D.K., Rieker, J.D., Leverson, V., Meyer, S., Meyer, S., 2006. Hydrologic  
642 Modeling Inventory: Cooperative Research Effort. *Journal of Irrigation and Drainage*  
643 *Engineering*. 132, 98–103. doi:10.1061/(ASCE)0733-9437(2006)132:2(98)

644 Singh, V.P., Woolhiser, D.A., 2002. Mathematical Modeling of Watershed Hydrology. *Journal*  
645 *of Hydrologic Engineering*. 7, 270–292. doi:10.1061/(ASCE)1084-0699(2002)7:4(270)

646 Tarboton, D., Idaszak, R., Horsburgh, J., Heard, J., Ames, D., Goodall, J., Band, L., Merwade,

647 V., 2014. A Resource Centric Approach For Advancing Collaboration Through Hydrologic  
648 Data And Model Sharing. International Conference on Hydroinformatics. CUNY Academic  
649 Works. [http://academicworks.cuny.edu/cc\\_conf\\_hic/314](http://academicworks.cuny.edu/cc_conf_hic/314).

650 Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Ames, D., Goodall, J.L., Band, L.E., Merwade, V.,  
651 Couch, A., Arrigo, J., Hooper, R.P., Valentine, D.W., Maidment, D.R., 2013. HydroShare:  
652 An online, collaborative environment for the sharing of hydrologic data and models  
653 (Invited). Present. 2013 Fall Meet. San Fr. Calif., 9-13 Dec.

654 Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodball, J.L., Merwade, V.,  
655 Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D.R., 2014. HydroShare:  
656 Advancing Collaboration through Hydrologic Data and Model Sharing, in: Ames, D.P.,  
657 Quinn, N.W.T., Rizzoli, A.E. (Eds.), International Environmental Modelling and Software  
658 Society (iEMSs) 7th International Congress on Environmental Modelling and Software.  
659 doi:978-88-9035-744-2

660 Taylor, P., Cox, S., Walker, G., Valentine, D., & Sheahan, P., 2014. WaterML2. 0: development  
661 of an open standard for hydrological time-series data exchange. Journal of  
662 Hydroinformatics, 16, 2. 425-446.

663 Winchell, M., R. Srinivasan, M. Di Luzio, and J. G. Arnold 2007, ArcSWAT interface for  
664 SWAT2005 - User's guide, Blackland Research Center, Texas Agricultural Experiment  
665 Station and Grassland, Soil and Water Research Laboratory, USDA Agricultural Research  
666 Service, Temple, TX.

667 Wosniok, C., Lehfeldt, R., 2013. A metadata-driven management system for numerical  
668 modeling. In Proceedings of OCEANS '13 MTS/IEEE, San Diego, CA, September 23–26.  
669