This is an Accepted Manuscript of an article published in Environmental Modelling and Software in July 2017 available online: http://dx.doi.org/10.1016/j.envsoft.2017.02.028

Design of a Metadata Framework for Hydrologic Models with an Example Application in HydroShare

Mohamed M. Morsy[a,b], Jonathan L. Goodall[a*], Anthony M. Castronova[c], Pabitra Dash[c], Venkatesh Merwade[d], Jeffrey M. Sadler[a], Md. Adnan Rajib[d], Jeffery S. Horsburgh[c], David G. Tarboton[c]

[a] Department of Civil and Environmental Engineering, University of Virginia, 351 McCormick Road, PO Box 400742, Charlottesville, VA, 22908, USA

[b] Irrigation and Hydraulics Department, Faculty of Engineering, Cairo University, P.O. Box 12211, Giza 12613, Egypt

[c] Utah Water Research Laboratory, Utah State University, 8200 Old Main Hill, Logan, UT 84322-8200, USA

[d] School of Civil Engineering, Purdue University, 550 Stadium Mall Drive, West Lafayette, IN 47907, USA

Highlights:

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

Abstract

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

Keywords:

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

Software Avaialbility:

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

1. Introduction

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

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

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

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

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

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

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

A motivating factor for this research is the design and development of a new system called HydroShare (https://www.hydroshare.org). The goal of HydroShare is to advance 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.

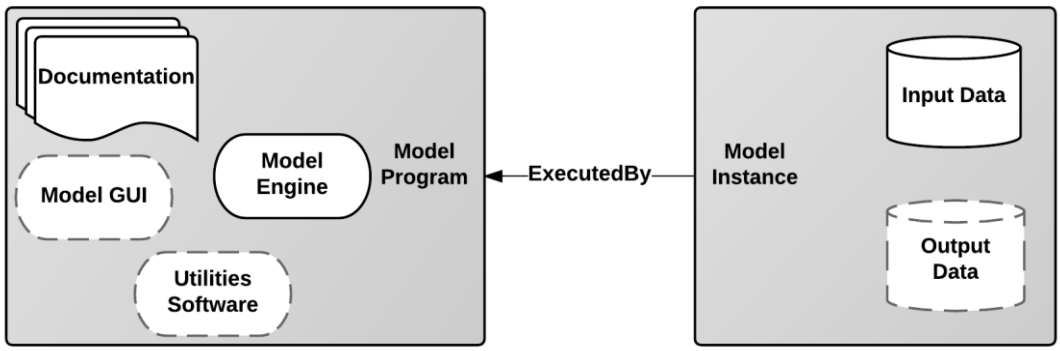


Figure 1. Key components of the model program and model instance resources.

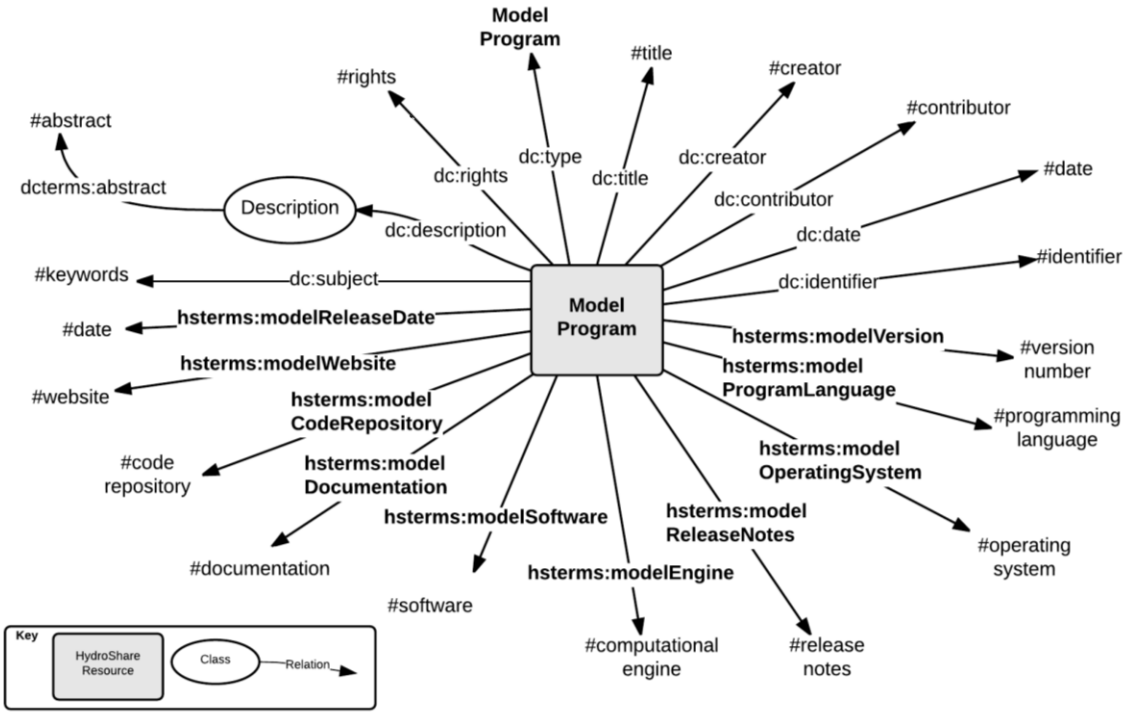
2.1.1. Model Program Resource Metadata

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

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

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

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



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.

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

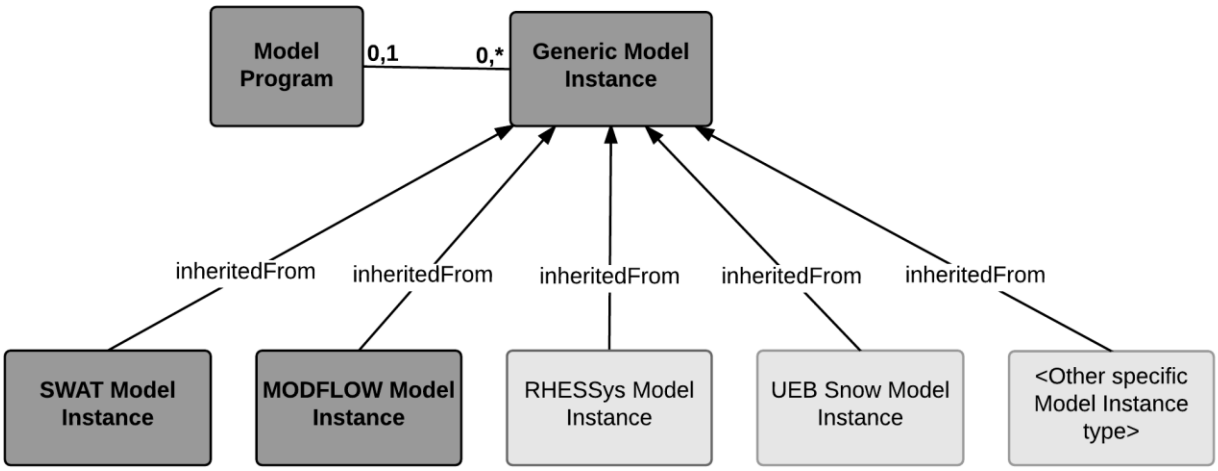
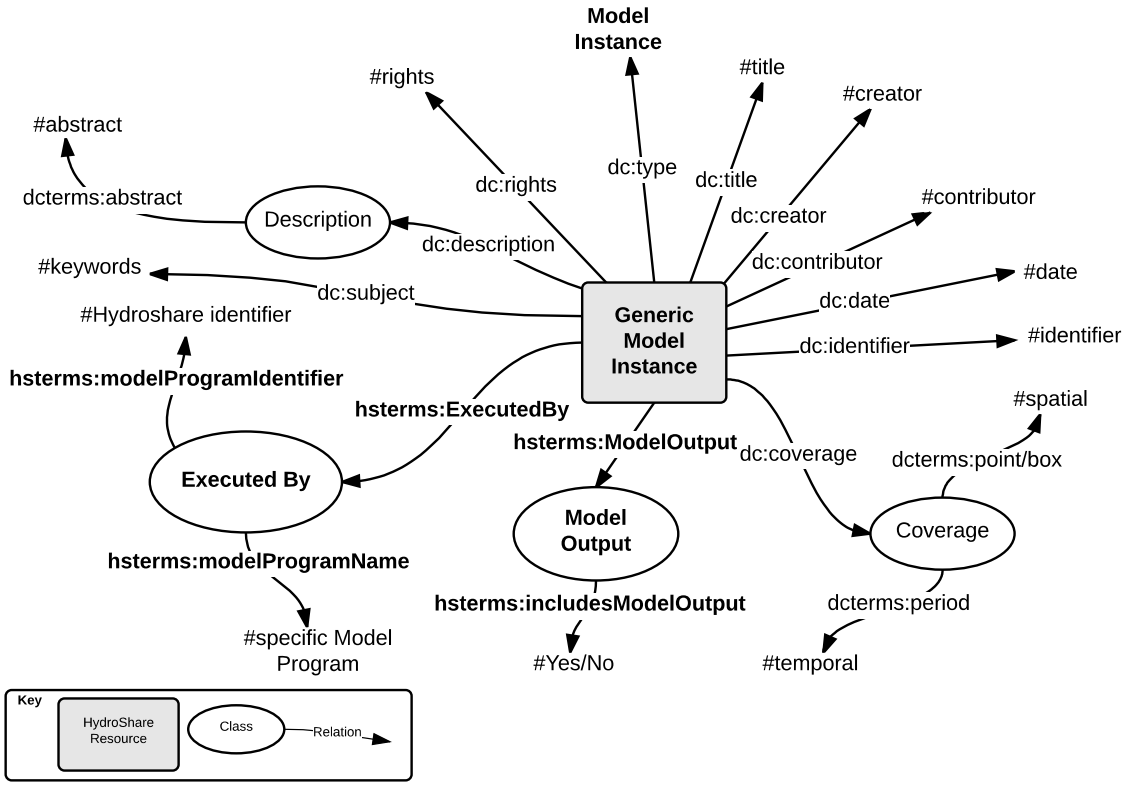


Figure 3. Generic model instance and specific model instance hierarchy. Model program, generic model instance, SWAT model instance, and MODFLOW model instance metadata have already been designed, while metadata for the other specific model instances are either in development or planned for the near future.

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

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

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



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
hsterms:modelOutput		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
hsterms:executedBy		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

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

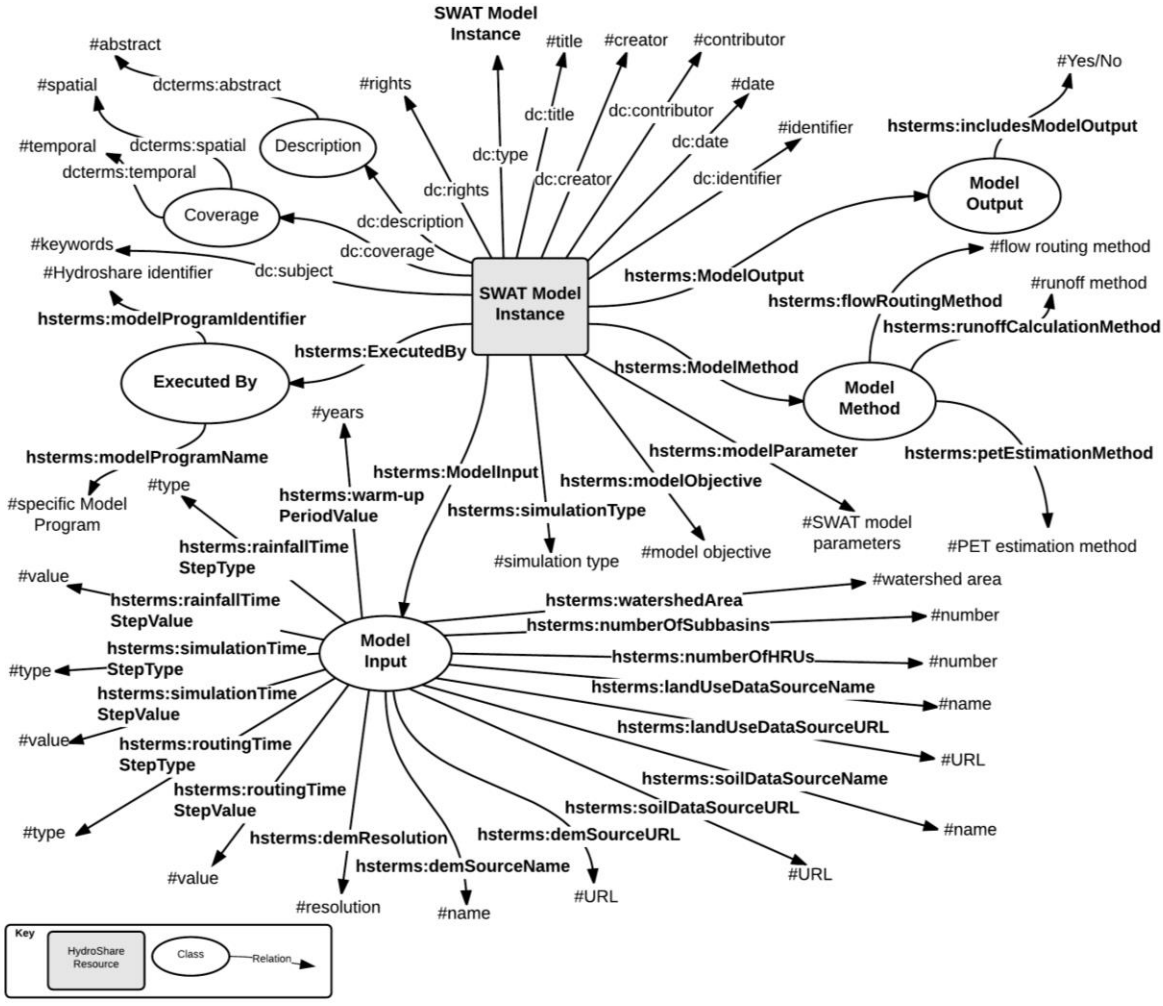


Figure 5. SWAT model instance metadata as RDF triples.

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)
hsterms:modelInput		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 km²
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
hsterms:modelMethod		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, tilage operation, inlet of draining watershed, irrigation operation, etc.)

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

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

2.2. Experimental Use Case

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

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

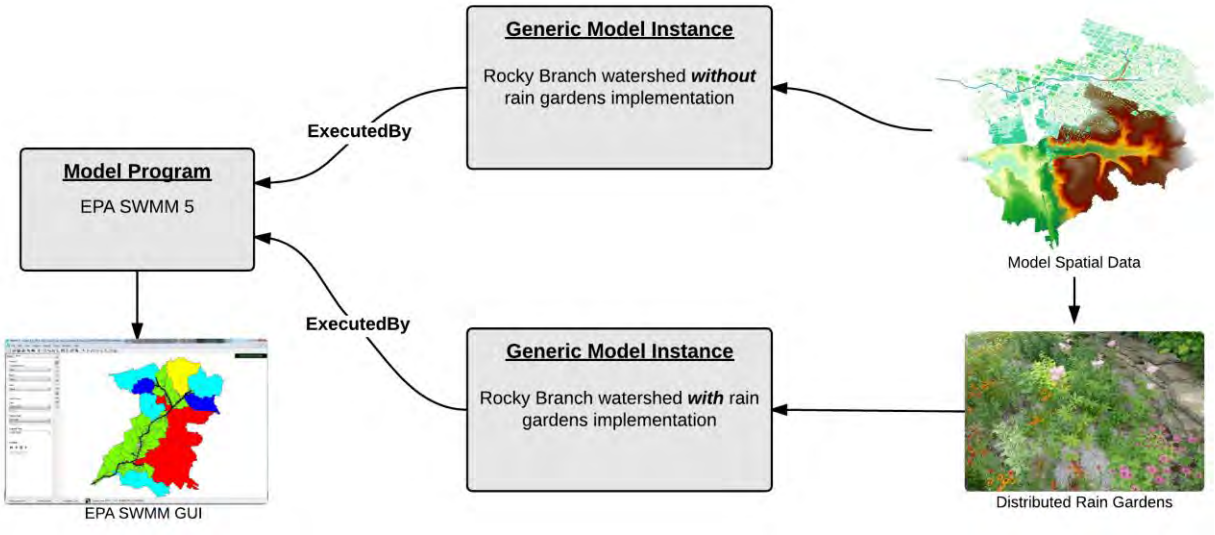


Figure 6. Use case implementation as a model program and two model instance resource types.

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

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

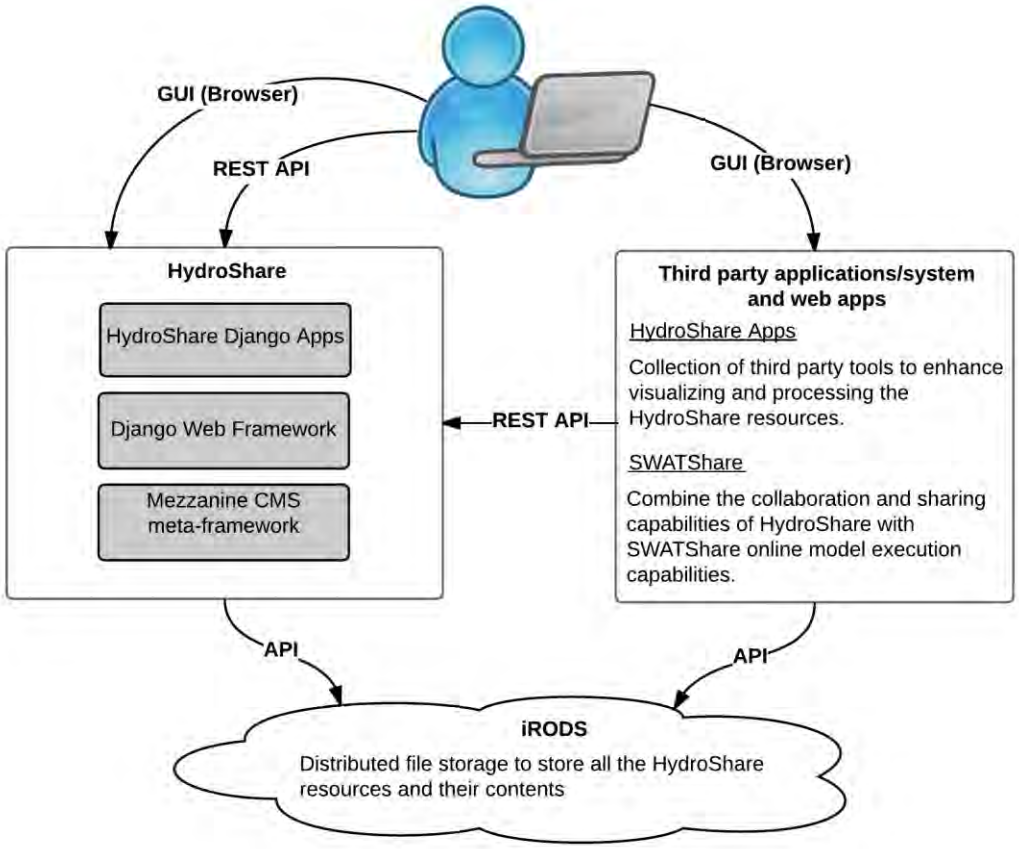


Figure 7. HydroShare's general architecture emphasizing the connections between the user, HydroShare, iRODS, and third party applications

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

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

3. Results

3.1. Results for Software Implementation within HydroShare

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

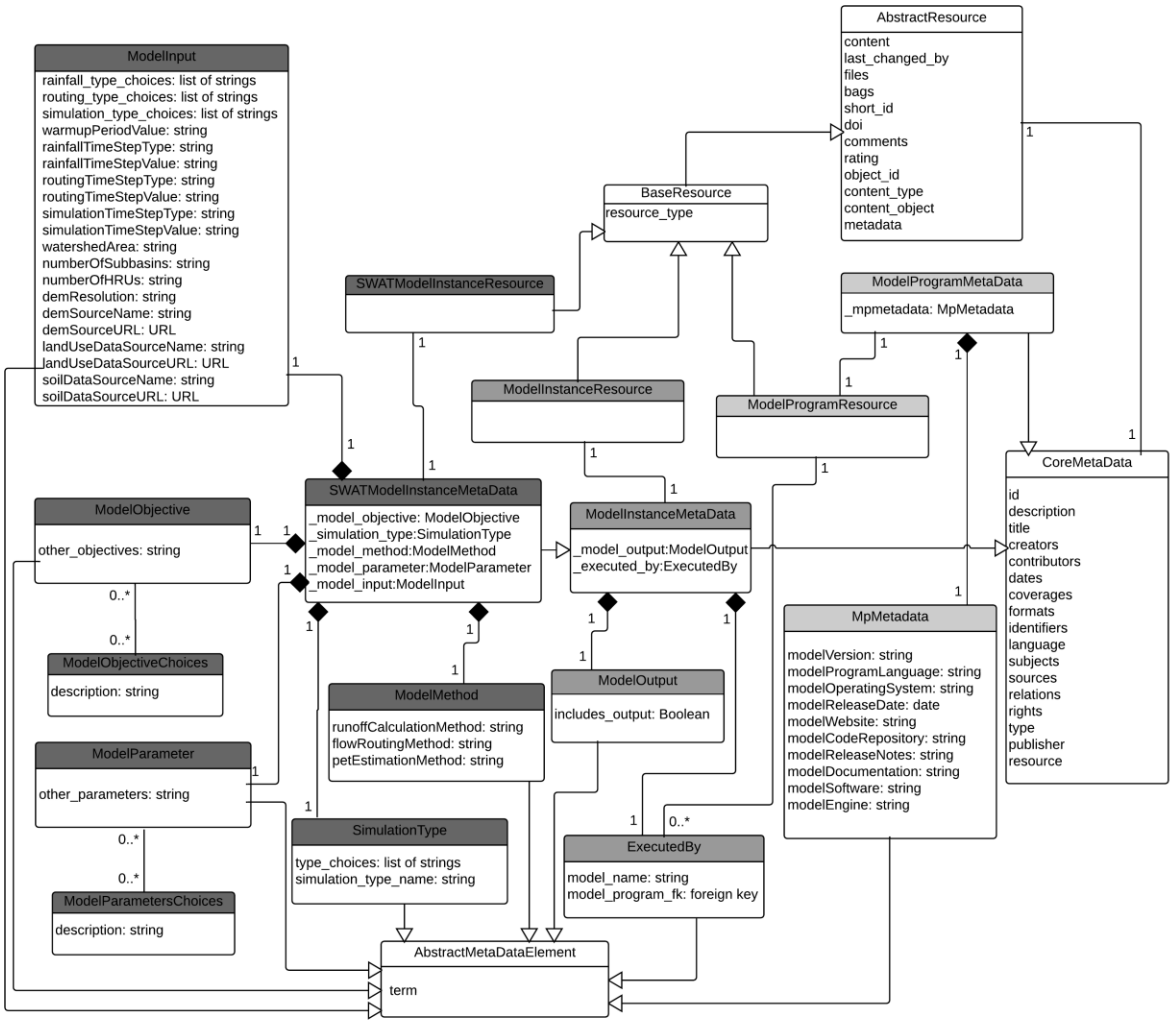


Figure 8. Metadata classes for model resources implemented within HydroShare.

Each Model resource type extends the BaseResource class by representing specific metadata elements as individual classes. These extended metadata classes inherit from the AbstractMetaDataElement class. In this class, there is one required attribute: term. Other attributes needed for further description can be added. For example, the extended metadata class ExecutedBy for the ModelInstance resource has the model_name, and model_program_fk attributes. The specific metadata elements are grouped in the CoreMetaData class. The ModelProgramMetaData, and ModelInstanceMetaData classes inherit from the CoreMetaData

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

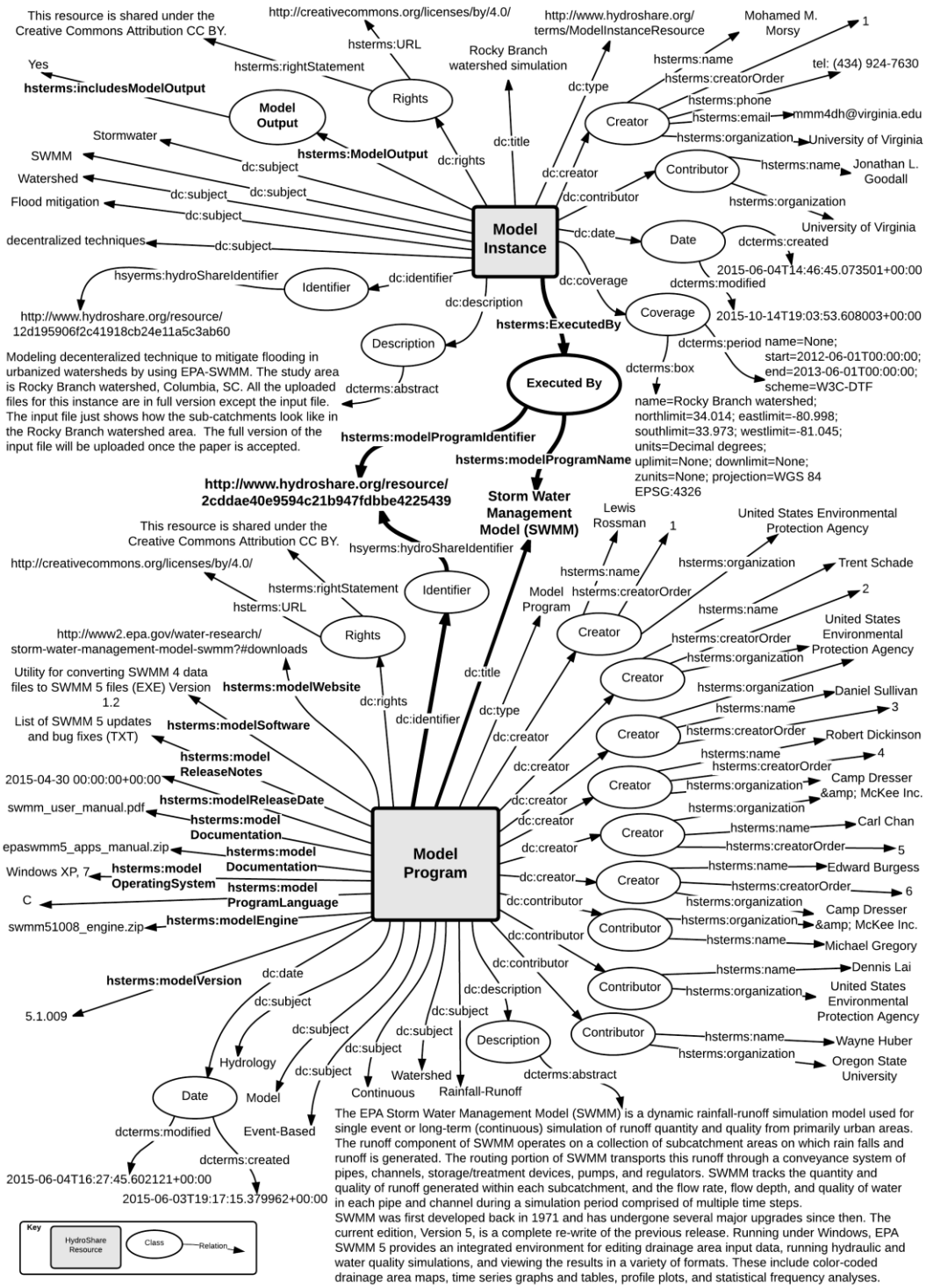
An important method of the CoreMetaData, ModelProgramMetaData, ModelInstanceMetaData, and SWATModelInstanceMetaData is get_xml. This method converts the stored metadata into an RDF-XML format. The CoreMetaData.get_xml method extracts the generic metadata elements, while the get_xml method for each specific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

3.2. Results from the Example Use Case

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

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

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

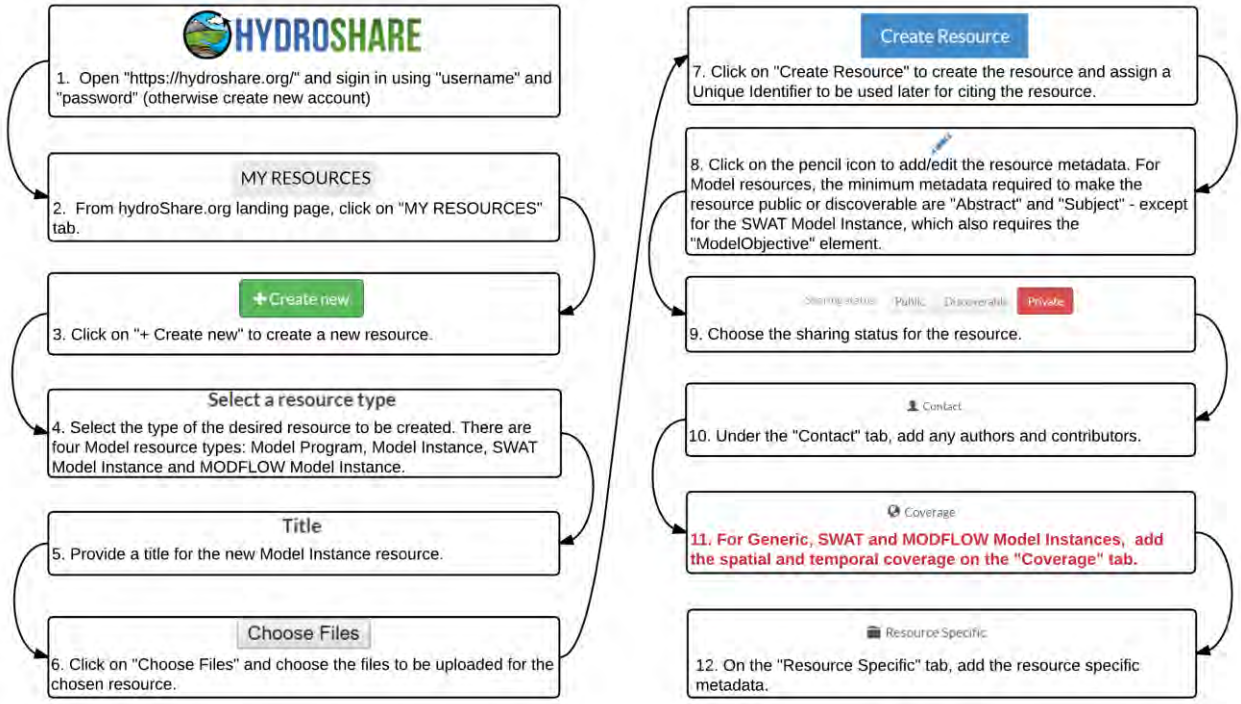


385

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

387 use case.

388



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.

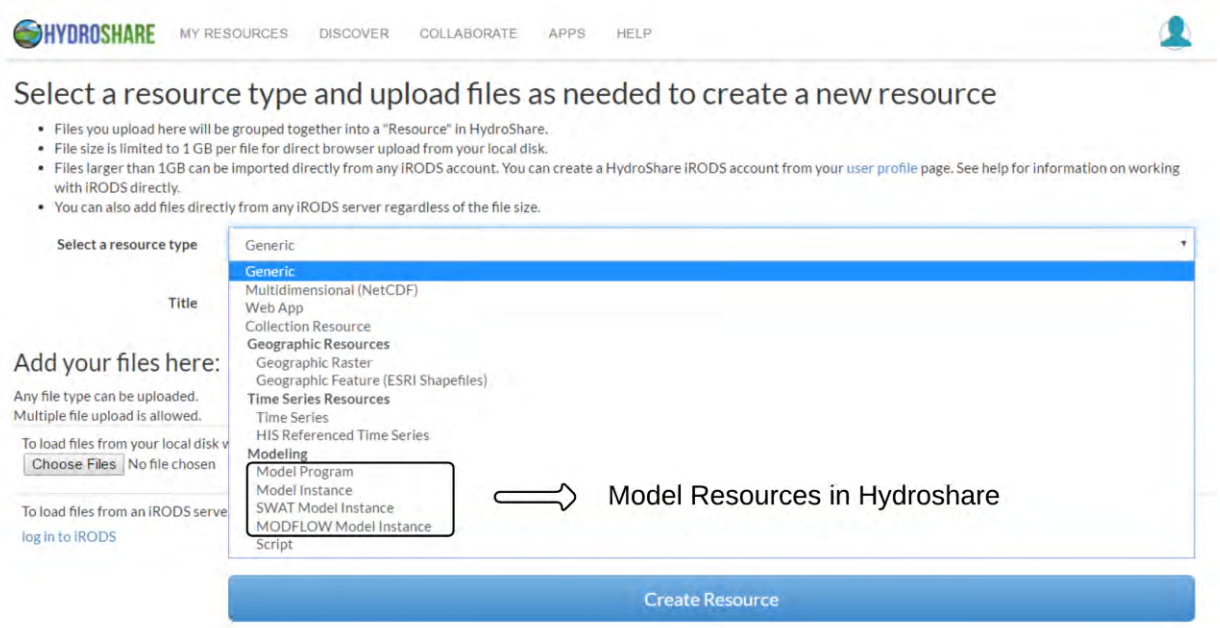
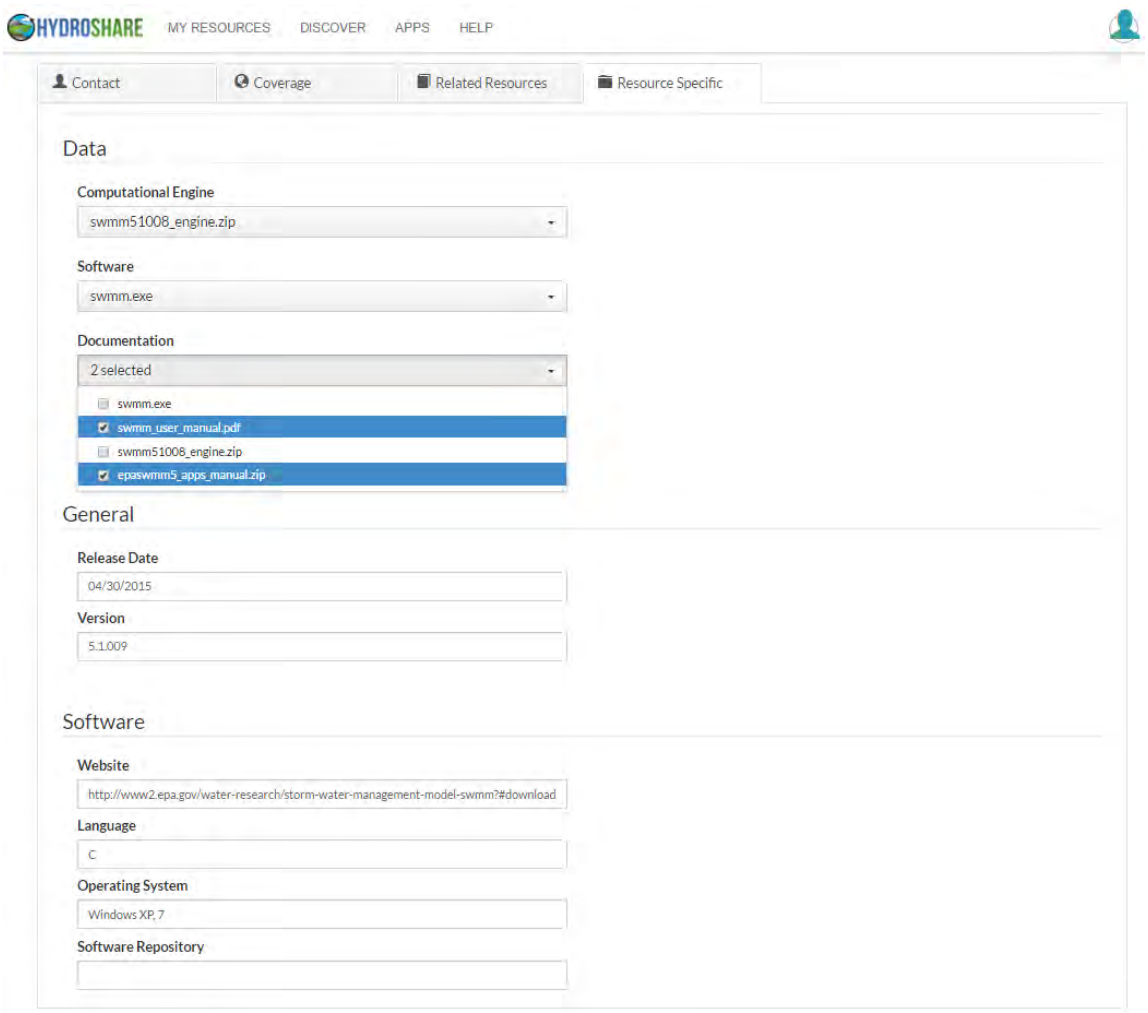


Figure 11. Screen shot showing model resource types currently implemented on hydroshare.org.

Figures 12 and 13 show the resource specific metadata for the model program resource and the generic model instance resource types, respectively, on their landing pages in HydroShare. These figures show HydroShare's metadata "edit" mode to illustrate all of the available metadata elements, as HydroShare's default is to hide metadata elements for which there are no values in regular "view" mode. Note that the model instance is linked to the model program used for execution (Figure 13). Under the "Model Program used for execution" heading on the generic model instance landing page, there is a dropdown list that collects all the available public model program resources in HydroShare. The user chooses the model program resource used to execute the model instance resource from the dropdown list (or adds a new model program resource if it is not already available). Once the user chooses the desired model program resource, a summary of the model program metadata is displayed to aid the user in confirming that the correct model program was selected.

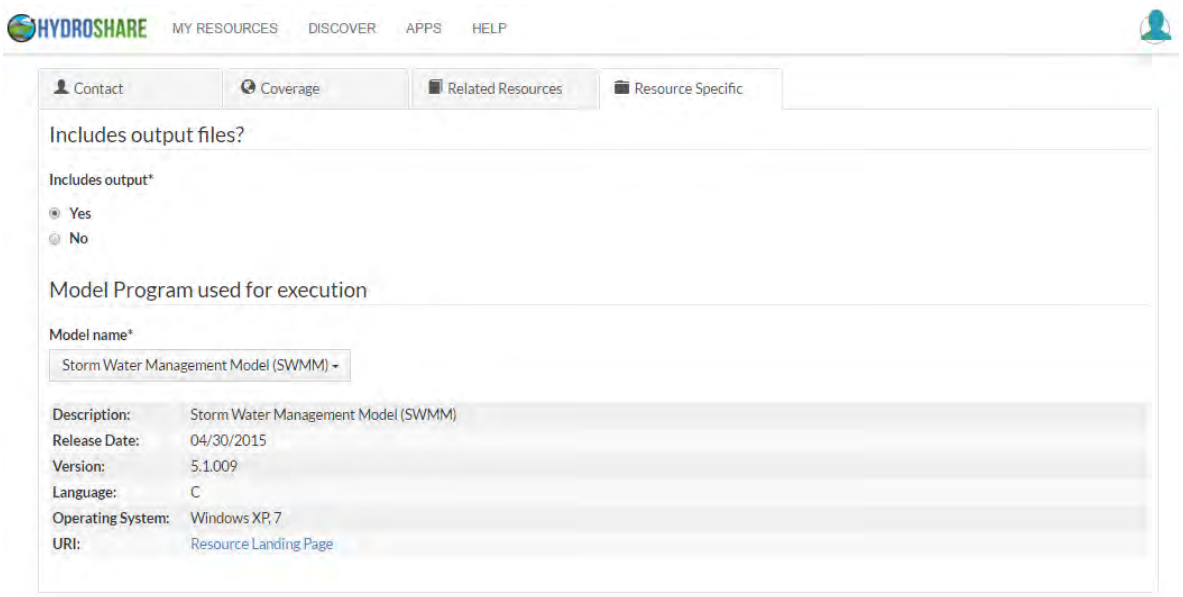
Another important aspect of the model instance resource is the coverage metadata. Figure 14 shows how the coverage metadata appears in the Coverage tab on the resource's landing page. As explained above, there are two types of coverage metadata elements: spatial and temporal. All of the spatial metadata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



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

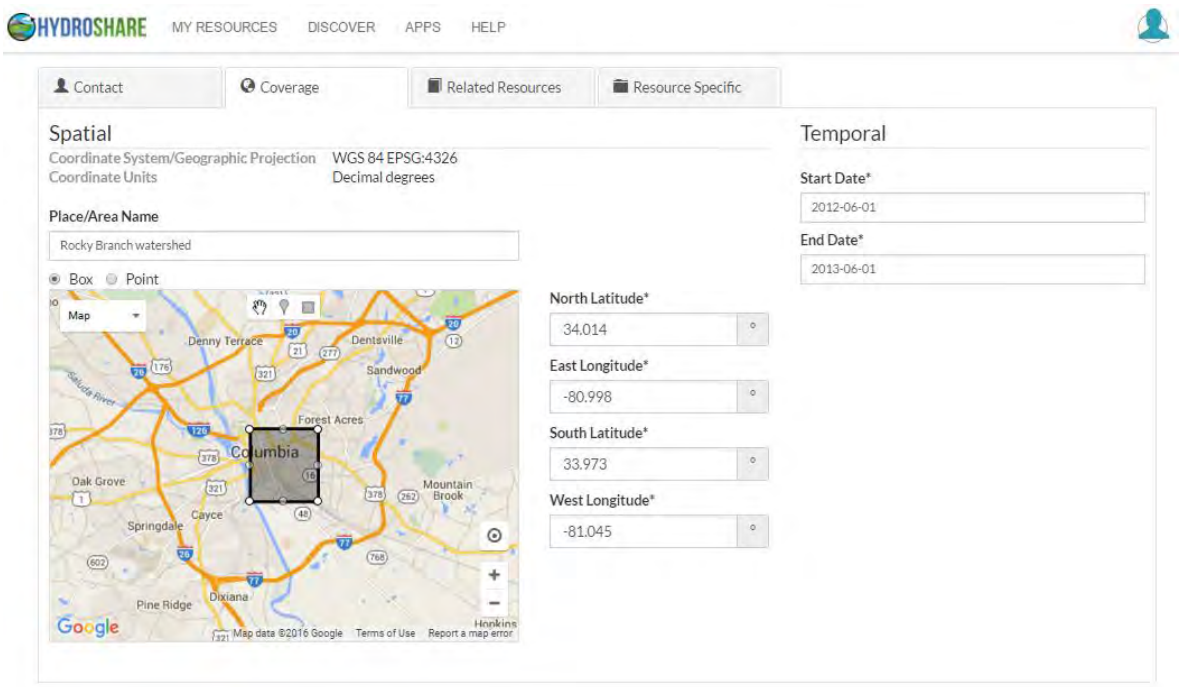


Figure 14. Model instance resource type coverage metadata on the resource's landing page (shown in edit mode) on hydroshare.org.

4. Discussion

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

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

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

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

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

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

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

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

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

5. Conclusions

This work presents a model metadata framework to support discovery, sharing and interpretation of hydrologic models. Key features of the framework are (1) that the model program and model instance are separate concepts with a one-to-many relationship (many instances may exist for a single model program), (2) that metadata for these concepts extend the 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 easy 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

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

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

6. Acknowledgements

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

7. References

Argent, R.M., 2004. An overview of model integration for environmental applications - Components, frameworks and semantics. Environmental Modelling and Software. 19, 219–234. doi:10.1016/S1364-8152(03)00150-6

Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environmental Modelling and Software. 78, 31–39. doi:10.1016/j.envsoft.2015.12.010

Boyko, A., J. Kunze, J. Littman, L. Madden, and B. Vargas, 2012. The BagIt File Packaging Format (v0.97), Network Working Group Internet Draft. http://tools.ietf.org/html/draft-kunze-bagit-10, accessed August 2016.

David, C.H., Famiglietti, J.S., Yang, Z.-L., Habets, F., Maidment, D.R., 2016. A decade of RAPID-Reflections on the development of an open source geoscience code. Earth and Space Science . 3, 226–244. doi:10.1002/2015EA000142

Elag, M., Goodall, J.L., 2013. An ontology for component-based models of water resource systems. Water Resoures Research. 49, 5077–5091. doi:10.1002/wrcr.20401

Essawy, B.T., Goodall, J.L., Xu, H., Rajasekar, A., Myers, J.D., Kugler, T.A., Billah, M.M., Whitton, M.C., Moore, R.W., 2016. Server-side workflow execution using data grid technology for reproducible analyses of data-intensive hydrologic systems. Earth and Space Science. 3, 163–175. doi:10.1002/2015EA000139

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 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.
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, 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, 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
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,

V., 2014. A Resource Centric Approach For Advancing Collaboration Through Hydrologic Data And Model Sharing. International Conference on Hydroinformatics. CUNY Academic Works. http://academicworks.cuny.edu/cc_conf_hic/314.

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

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

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

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

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