Advancing the Reproducibility of Hydrologic Models: An Extensible Model Metadata Framework and Cyberinfrastructure for Complex Modeling

A Dissertation

Presented to

The Faculty of the School of Engineering and Applied Sciences

University of Virginia

In Partial Fulfillment

of the requirements for the Degree of

Doctor of Philosophy (Civil Engineering)

By

Iman Maghami

August 3, 2022

Abstract

A variety of hydrologic computational models are used by domain scientists to address specific water challenges such as floods, droughts, and water pollution. The number of scientific studies making use of such diverse and often complex models has been rapidly increasing. Ideally, the investment in models and approaches used in these past studies could be reproduced, replicated, and leveraged in future studies. Many researchers in recent years have emphasized reproducibility as an essential part of scientific research. However, the reproducibility of model-based scientific studies is still lacking in large part due to the proliferation and complexity of hydrologic models. In this research, the components of these models are defined as two distinct concepts: the modeling software referred to as a *model program* and the model input and output files referred to as a *model instance*. The overall goal of the research reported in this dissertation is to foster the reproducibility of model-based hydrologic studies through (i) creating a metadata framework to describe model programs and model instances in a more extensible way and (ii) designing cyberinfrastructure to overcome challenges in reproducing large-scale computational studies. The first part of this research goal leads to the first two studies in this dissertation. First, an extensible hydrologic model metadata framework design using a user-defined metadata schema is presented. To overcome the fact that the properties of a model instance needed for a single hydrologic model program can vary significantly, I present a standardized way for encoding an extensible machine-readable model metadata schema that specifies the metadata elements for the model instances of any model program. Second, I present a methodology for building a model instance metadata schema for model programs and discuss strategies for how to manage the variety of metadata schemas that might be developed by the modeling community. The third study focuses on the second part of the overall goal of this dissertation research and aims to advance the cyberinfrastructure needed to overcome data management and software architecture challenges to reproduce large-scale computational studies. The goal is to make computational hydrologic studies easier to reproduce and reuse. Across the three studies, this research work advances reproducibility by demonstrating how model agnostic metadata frameworks can advance model sharing and reuse, how it is possible to create a generic model metadata framework for hydrologic models, how containerization of the model allows for portability across computing environments, how Globus can be used for large data transfer between scientific cloud services, and how Jupyter can provide a gateway to HPC environments. Results of this research provide scientists and engineers new ways to conduct computing and data-intensive modeling studies using modern cyberinfrastructure that fosters reproducibility. These results not only impact the hydrologic modeling field, but can be broadly adopted by environmental and other scientific disciplines that follow a similar practice of making use of model program and model instance objects in their research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The need for research reproducibility, the ability of a researcher to duplicate and verify the findings of a prior study, has been emphasized by numerous studies (Choi et al., 2021d; Essawy et al., 2018; Horsburgh et al., 2016; Nüst et al., 2018; Piccolo and Frampton, 2016; Stodden et al., 2013). Some studies have assessed the research reproducibility in different scientific disciplines (e.g., Baker, 2016; Obels et al., 2020; Stagge et al., 2019) and found what has been called a "reproducibility crisis". Within the field of hydrology and water resources, Stagge et al. (2019) investigated 360 articles in six leading journals to examine their online data availability and reproducibility of study results. The result of their study indicated that only 5.6% of the articles had data, model code, and directions for use available online, and only results of 1.1% were completely reproducible while 0.6% were partially reproducible.

Even the word "reproducible" has different meanings and definitions in the modeling literature. Essawy et al. (2020) defined a taxonomy for levels of reproducibility for modelling studies that comprise a progression of increasing time and effort. These levels of model reproducibility begin with repeatability then runnability, reproducibility and, finally, replicability. The two lowest levels, repeatability and runnability, are performed by the original researcher while the highest two levels, reproducibility and replicability, are performed by a new researcher attempting to reproduce someone else's work. In this work, I mainly deal with reproducibility and replicability levels. According to Essawy et al. (2020), the third level, reproducibility, is obtaining consistent results using the same input data, computational steps, methods, code and conditions of analysis by a new researcher on a new machine. The fourth level, i.e., replicability, is obtaining consistent results across studies aimed at answering the same scientific question with new data. While I acknowledge these levels of reproducibility exist, when using the term "reproducibility" in this work, I most often mean the two highest levels defined by Essawy et al. (2020), i.e. "reproducibility" and "replicability." That said, it is not often necessary to precisely define what level of reproducibility I am referring to in the work. Therefore, most often the general term "reproducibility" is used but, when it is necessary, I will use the Essawy et al. (2020) taxonomy to distinguish between levels of reproducibility.

Although several possible reasons can cause the previously mentioned inability to reproduce past studies, reviewing recent studies reveals an ongoing need to extend the cyberinfrastructure for the sharing, discovering, and reuse of data and models build by others by introducing novel approaches to foster the reproducibility of model-based studies (e.g., Choi et al., 2021g; Essawy et al., 2020; Morsy et al., 2017; Riddick et al., 2020). The objective of this dissertation is to improve the reproducibility of model-based hydrologic studies by achieving three distinct research goals: (1) designing and implementing a general model metadata framework, (2) implementing general metadata design rules to design a model metadata schema for a specific environmental model, and (3) presenting cyberinfrastructure for overcoming reproducibility barriers in data and computationally intensive hydrologic modeling utilizing a strategy and architecture presented by previous researchers.

Metadata, defined as "structured data about data" (Duval et al., 2002), is a crucial component of any digital information system and helps people and machines find, access, and use information (Greenberg et al., 2013). Without metadata standards, information systems lack the ability to properly interpret and contextualize the data. This can lead to problems for end users including an inability to search relevant data, an inability to find related information, and an inability to operate on data for deriving new information. Adopting standard metadata is a widely acknowledged way to advance reproducibility (Essawy et al., 2020; Leipzig et al., 2021; Qin et al., 2016). While there are many well-established metadata for data, fewer studies have started to adopt, design, and frame agreed-on metadata for models (Morsy et al., 2017).

For numerical models, one example of metadata to describe a model that contains the input and output data for model scenarios was done by Hill et al. (2001). They provided approximately 165 metadata elements divided into ten sections to help find and reuse models (http://www.dlib.org/dlib/june01/hill/appendix.html). Lehfeldt (2013) proposed a concept for metadata-driven architecture to describe computational fluid dynamics simulations as well as a method to integrate model descriptions into spatial data infrastructures. The Community Surface Dynamics Modeling System (CSDMS) (Overeem et al., 2013; Peckham, 2014) created a metadata framework and use it to describe over 420 geoscience models and tools, including over 100 hydrologic models within its portal (http://csdms.colorado.edu). While the CSDMS model metadata focuses on the software for executing a model, it does not extend to the input/output files for a specific model simulation as it is primarily a portal for sharing the model software and not the input/output files associated with running a specific instance of the model software. The OntoSoft project (Gil et al., 2016, 2015), as part of the National Science Foundation EarthCube Initiative, frames an ontology and provides a portal to capture metadata for scientific software in a formal way. The focus of the metadata captured by the OntoSoft ontology is on the knowledge needed for sharing and reuse of "software", broadly used by them to refer to scripts as well as more complex software such as modeling software (Essawy et al., 2017).

While this past research in model metadata started to set the stage to adopt, design, and frame agreed-on metadata for models and contributed a useful metadata design relevant to hydrologic models, perhaps the most related prior work to this project is the work of Morsy et al. (2017). In this work, models are defined as two distinct concepts, i.e., modeling software (called the model program, and referred to as "MP" in this research) and the model input file(s), that are fed to a modeling software (called the model instance, and referred to as "MI" in this research) to conduct a model simulation (Morsy et al., 2017). A MP includes software developed for running a model simulation and generating outputs, while a MI is the application of the MP to a specific study area which includes the input files and, optionally, the output files for a particular simulation. Morsy et al. (2017), who initially established these two objects, designed a generic metadata framework for hydrological and environmental models based on the conceptualization of model components as MP and MI.

The Morsy et al. (2017) approach was to extend the Dublin Core metadata framework to be applied to a wide range of hydrological and environmental models. They also added an extension to the generic model program metadata framework for two of the most used hydrological models: SWAT (Soil and Water Assessment Tool) and MODFLOW (Modular Three-Dimensional Finite-Difference Groundwater Flow Model). Their model-specific extended metadata helped users describe the SWAT and MODFLOW MIs with better details using a pre-defined metadata schema specific to these two models. They implemented their design in the web-based collaborative system, HydroShare (https://www.hydroshare.org/). HydroShare is designed to advance sharing, accessing, and discovering hydrologic and environmental data and models

(Tarboton et al., 2014b, 2014a). In their implementation on HydroShare, the users could upload their data and/or model files and their corresponding metadata as a “resource”, the basic digital content unit in HydroShare. They implemented four different model resources (a MP, and generic, non-model specific MI, a SWAT MI, and a MODFLOW MI) on HydroShare. Their design was one of the existing best practices for model metadata and inspired future studies (e.g., Riddick et al., 2020), making a significant step towards a more established metadata framework for environmental models.

Despite its success, one challenge with the Morsy et al. (2017) design is that the properties of a MI for one piece of hydrologic and environmental modeling software (i.e., MP) can vary significantly from another. This variety necessitates using a customizable metadata schema to describe MIs for any given MP. To this point, supporting specific MPs and their accompanying MIs within a hydrologic information system has been laborious, requiring repository developers to create and maintain individual metadata schemas manually. As a result, only a limited number of specific MPs (i.e., SWAT and MOFLOW) have been supported by the HydroShare system. What is needed is a more general and flexible approach for defining model-specific metadata. This knowledge gap leads to the first two studies reported in this dissertation.

The first study aims to present an extensible environmental model metadata framework using a user-defined metadata schema. Therefore, a standardized and extensible way for encoding a machine-readable model metadata schema file that specifies the metadata elements for MIs of a MP is proposed. This standard metadata schema format enables data repository users, the modelers themselves, to define the metadata fields for any arbitrary MP. The design, building up on the previous design by Morsy et al. (2017), is implemented in HydroShare.

The second study, building on the first objective, aims to explore building model instance metadata schemas for arbitrary model programs and discusses how to manage the variety of metadata schemas that might be developed by the modeling community. To achieve this, the metadata elements that need to be standardized across hydrologic and environmental models are identified and an example metadata schema is built for a particular hydrologic model, i.e., Structure for Unifying Multiple Modeling Alternative (SUMMA) (Clark et al., 2015b), use case. Then, an example metadata schema is built for a particular hydrologic model (Structure for SUMMA use case. Upon creation of a number of different metadata schemas for different models by modelers, there will be a need to organize the metadata schemas so that they can be easily shared, found and reused. Therefore, a method for organizing the metadata schemas is discussed to advance the reportability and transparency.

Reproducibility goes beyond sharing code and data as open-source and online resources described by metadata. The code and data must be accompanied by well-documented workflows with readable and reusable code (Chen et al., 2020). Choi et al. (2021d), described a general strategy for creating modern cyberinfrastructure to support open and reproducible hydrologic modeling. This approach integrates three main components: (1) online data repositories; (2) computational environments leveraging containerization and self-documented computational notebooks; and (3) Application Programming Interfaces (APIs) that provide programmatic control of complex computational models. Choi et al. (2021d) focused mainly on the system design and demonstrated their approach with a fairly simple modeling example. However, reproducibility in computational hydrology can present some difficult challenges when dealing with large-scale hydrologic studies (Hutton et al., 2016). These challenges mostly pertain to the use of “big data” and computationally expensive and time-consuming resources needed for reproducibility of complex hydrologic modeling studies. Hutton et al. (2016) notes that in these cases, new techniques are needed to ensure scientific rigor. This knowledge gap leads to the third study reported in this dissertation.

The third study aims to provide an example of the overall system design outlined by Choi et al. (2021d) as applied to a large-scale hydrologic study by Van Beusekom et al. (2022). To ensure reproducibility of large-scale hydrologic studies, the necessary cyberinfrastructure to reproduce Van Beusekom et al. (2022) study for selected sub-domains are developed and the embedded challenges and opportunities are discussed. The developed cyberinfrastructure utilizes software components to enable intuitive and online access to computational environments, including high performance computing (HPC). Using HydroShare as the data repository and containerization of the pySUMMA API along with a computational gateway interface of Jupyter IPython notebooks hosted on the CyberGIS-Jupyter for Water (CJW), it was possible to overcome data management and software architecture challenges to reproduce large-scale computational studies. A model workflow run-time performance analysis was conducted to evaluate the effectiveness of the computational environments to reproduce different-sized model simulations.

The remaining chapters of this dissertation are organized as follows. Chapters 2-4 are results from the three studies described in this chapter, each written as a standalone study to facilitation their publication in peer-reviewed journals. Chapter 5 seeks to tie these three studies together by presenting overall conclusions drawn across the three studies.

This chapter is in preparation for submission to a peer-reviewed journal. Current coauthors of the study are Iman Maghami, Jonathan L. Goodall, Mohamed M. Morsy, Jeffrey M. Sadler, Jeffery S. Horsburgh, Pabitra K. Dash, Youngdon Choi, Kevin Chen, Martin Seul, Scott Black, and David G. Tarboton.

Chapter 2

An Extensible Environmental Model Metadata Framework Using a User-defined Metadata Schema

2.1 Introduction

Domain scientists and engineers use a broad range of hydrologic and environmental models to address and understand various environmental and water resource issues such as climate change, floods, droughts, and water pollution (Addor and Melsen, 2019; Tayfur, 2017). The number of scientific studies making use of such diverse and often complex models has been rapidly increasing. Ideally, the investment in models and approaches used by existing studies can also be leveraged by future studies. Many researchers in recent years have studied this idea of reproducibility as an essential part of scientific research (e.g., Choi et al., 2021; Essawy et al., 2020, 2018; Horsburgh et al., 2016; Hutton et al., 2017; Morsy et al., 2017; Stagge et al., 2019). However, the reproducibility of scientific studies still needs to be improved given the recent rapid development of models.

Some studies have assessed research reproducibility in different scientific disciplines (e.g., Baker, 2016; Obels et al., 2020; Stagge et al., 2019). Baker (2016) surveyed 1576 researchers across various scientific disciplines and revealed that more than 70% of the surveyed researchers tried but failed to reproduce studies published by other researchers. In another similar study, but only focused on the field of hydrology and water resources, Stagge et al. (2019) estimated with 95% confidence that the results of only 0.6% to 6.8% out of a sample of 1989 articles published in water resources-related journals during the year of 2017 could be reproduced. According to these studies, methods, codes, and raw data not available from the original studies were among the main contributing factors to the irreproducibility of scientific studies. Given that, novel approaches for sharing, discovering, and reusing data and models built by others have been proposed in recent years to improve the reproducibility of scientific research.

Adopting standard metadata is a widely accepted way to strengthen the computational resources for sharing, discovering, and reusability of research (Essawy et al., 2020; Horsburgh et al., 2016; Leipzig et al., 2021; Morsy et al., 2017). While much effort has been invested in establishing metadata standards and profiles for scientific data, fewer efforts have been made to adapt, design, and frame agreed-on metadata for models (e.g., Hill et al. (2001), Wosniok and Lehfeldt. (2013), the Community Surface Dynamics Modeling System (CSDSM) (http://csdms.colorado.edu/), and OntoSoft (Gil et al., 2016, 2015)). Yet, such metadata are necessary for discovering, accessing, and reusing models and are essential for reproducing scientific studies where models are employed.

Morsy et al. (2017) designed a generic metadata framework for hydrological and environmental models. This generic metadata framework extended the Dublin Core metadata framework ("Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1.," 2012) to be applied to a wide range of hydrological and environmental models. They defined computational model components as two distinct concepts, i.e., model program (referred to as "MP" in this paper) and model instance (referred to as "MI" in this paper). A MP resource included software developed for running a model simulation and generating outputs. In contrast, a MI resource was the application of the MP to a specific spatial and temporal domain. Therefore, a MI resource included the input files and, optionally, the output files for a particular simulation. They also added an extension to the generic MP metadata framework for two popular and frequently used hydrological models, i.e., the Soil and Water Assessment Tool (SWAT) and the Modular Three-Dimensional Finite-Difference Groundwater Flow Model (MODFLOW). This extended metadata helped users describe SWAT and MODFLOW MIs with better details using a pre-defined metadata schema specific to those two models. They implemented their design in the web-based collaboration and repository system, HydroShare (https://www.hydroshare.org/). HydroShare was designed to advance sharing, accessing, and discovering hydrologic and environmental data and models (Tarboton et al., 2014b, 2014c). HydroShare users upload their data and/or model files and provide their corresponding metadata as a "resource", the basic digital content unit in HydroShare, and sharing MPs and MIs as resources in HydroShare enhances reproducibility because researchers can download and execute the models to verify, reproduce, and extend results. In Morsy et al.'s (2017) implementation in HydroShare, four different model resource types were defined, i.e., generic MP, generic MI, SWAT MI, and MODFLOW MI. Their design presented an outstanding practice for model metadata, has been used by recent studies (e.g., Riddick et al., 2020), and took significant steps towards a more established metadata framework for models.

However, although their approach highlighted a road toward sharing models developed by previous research in a reproducible way, their initial MP and MI metadata design assumed that a new HydroShare resource type would be defined for each new model. Although the metadata concepts were sound, the implementation approach ultimately lacked sufficient scalability and flexibility to be easily expanded and implemented for other model types. This was generally true of HydroShare's early approach to defining typed resources for different content types. This first approach only worked for a limited range of modelers, missing the wider range of environmental models that could easily benefit from it (i.e., the scalability problem), because their work only provided the specific (or extended) metadata support for two hydrologic models (SWAT, and MODFLOW) and most modelers lacked the specific skills needed to define a new HydroShare resource type,.

The limitations of Morsy et al.'s (2017) work were due to several challenges faced by the HydroShare team in providing a significant number of existing environmental models with specific metadata support. Their design was not flexible enough (i.e., the flexibility problem) to give the modelers complete ownership of model metadata for two main reasons. First, modelers could not edit which specific metadata elements were defined for the models that had particular metadata support (i.e., SWAT and MODFLOW). Definition of MP and MI resource types was done by the HydroShare development team and most modelers lack the web development skills needed to define a specific HydroShare MP or MI resource type. Second, although modelers could benefit from “Additional Metadata” as key-value pairs that could be used to add any arbitrary metadata to MP or MI resources in HydroShare (both models that had specific metadata support, i.e., SWAT, and MODFLOW, and models that had only generic metadata support, i.e., any model other than SWAT and MODFLOW), such “Additional Metadata” could only be applied to one MP or MI resource at a time. For example, if there were many MI resources with the same metadata executed by one specific MP resource, the modeler needed to repeat adding “Additional Metadata” for each MI resource, which could make this task overwhelming. Moreover, the “Additional Metadata” for adding arbitrary metadata being designed as key-value pairs of string type, could not support more enhanced metadata (e.g., letting the user choose the data type or choose values from a controlled vocabulary), and there was no guarantee that multiple modelers describing the same MP or MI resource type would use the same “Additional Metadata” elements.

Given these limitations and Hydroshare’s shift from multiple typed resources to a single resource type with typed content within a resource, the objective of this study was to improve and transfer the ability to provide support for the numerous MPs used by the hydrological and environmental science community - from HydroShare repository and tool developers to the modeling community. To achieve this goal, we designed and implemented a flexible and scalable method for specifying metadata for hydrological and environmental models by expanding on the previous approach (i.e., Morsy et al., 2017) to address the scalability and flexibility problems. The new design reported here is scalable in the sense that modelers, who are repository (e.g., HydroShare) users who create resources containing hydrologic and environmental models, can support their own MI with customized metadata defined and supported by themselves, i.e., modelers define and update the metadata for their models without any need for further effort from the repository and tool developers.

The new design is flexible as it introduces a novel and standardized way for encoding a machine-readable model metadata schema file that specifies the metadata elements for MIs of a MP. This metadata schema facilitates sharing and documenting of models and makes valuable strides towards enhancing reproducibility of hydrologic and environmental research. This is achieved by giving the ability to the repository users, the modelers themselves, who are expectedly more knowledgeable and invested in describing MIs than the repository and tool developers (e.g., HydroShare developers), to create the metadata schema by specifying the metadata elements that best describe MIs of a MP. Using schema-based metadata, elements can be defined and new elements added without additional effort by repository developers compared to Morsy et al.’s (2017) design. Moreover, the implementation example we demonstrate using HydroShare shows how using a standardized, machine-readable, model metadata schema file makes this approach suitable for any arbitrary model. This means that in the new design, only one MI content type and one MP content type exist in HydroShare, which simplifies HydroShare’s codebase while still solving the scalability and flexibility problems. The MP and MI content types introduced by this study can support describing any specific model.

This paper describes the design of the flexible and scalable metadata schema and demonstrates its use. Section 2 discusses the design of the model metadata framework (MI content type aggregation, MP content type aggregation, and MI metadata schema). Later in that section, we employ a use case of a hydrological modeling study using a hydrological model (SWAT) to test and validate the implemented design in HydroShare. Section 3 presents the implemented software in HydroShare and the results from the example use case. Finally, we discuss the approach we implemented and provide the summary, conclusions, and possible future steps.

2.2 Methodology

2.2.1 Metadata framework design

As mentioned in the introduction section, this study enables transferring the responsibility to support the numerous MPs used by the hydrological science community - from the repository and tool developers to the modeling community – user-level responsibility. This was done by redesigning the metadata framework presented by Morsy et al. (2017) by focusing on addressing the limitations of their design, i.e., lack of flexibility and scalability through a standardized way of encoding machine-readable model metadata schema file specifying the metadata elements for MIs of an MP.

The core of the design by Morsy et al. (2017) was to consider a computational model as two distinct components - i.e., a MP resource (i.e., software and engine) and a MI resource (i.e., input and optionally output files). This separation was made to: first, let several MIs link to one specific version of a MP that could execute those MIs; and second, to avoid redundancy (if MI and MP were combined together in the same resource, the same MP would be stored several times with each MI executed by that MP (Figure 2.1, left side)). An “ExecutedBy” relation as a many-to-one to link between any number of instances and the MP used for execution of an MI was used in Morsy et al. (2017). Consistent with HydroShare’s method for encoding metadata, Morsy et al. (2017) used Resource Description Framework (RDF) to encode concepts and their associated metadata formally. RDF uses a subject, predicate, and object structure (http://www.w3.org/RDF). As a simple example, this basic structure could be used to show that a MI (subject) is executed by (predicate) a MP (object) (Figure 2.1. Left side). Each resource had core metadata defined by the Dublin Core metadata framework and extended metadata designed through their research that was encoded and stored on disk using RDF-XML. Additionally, their design included four different model resources types: generic MP resource, and generic MI resource, SWAT MI resource, and MODFLOW MI resource (Figure 2.1, left side).

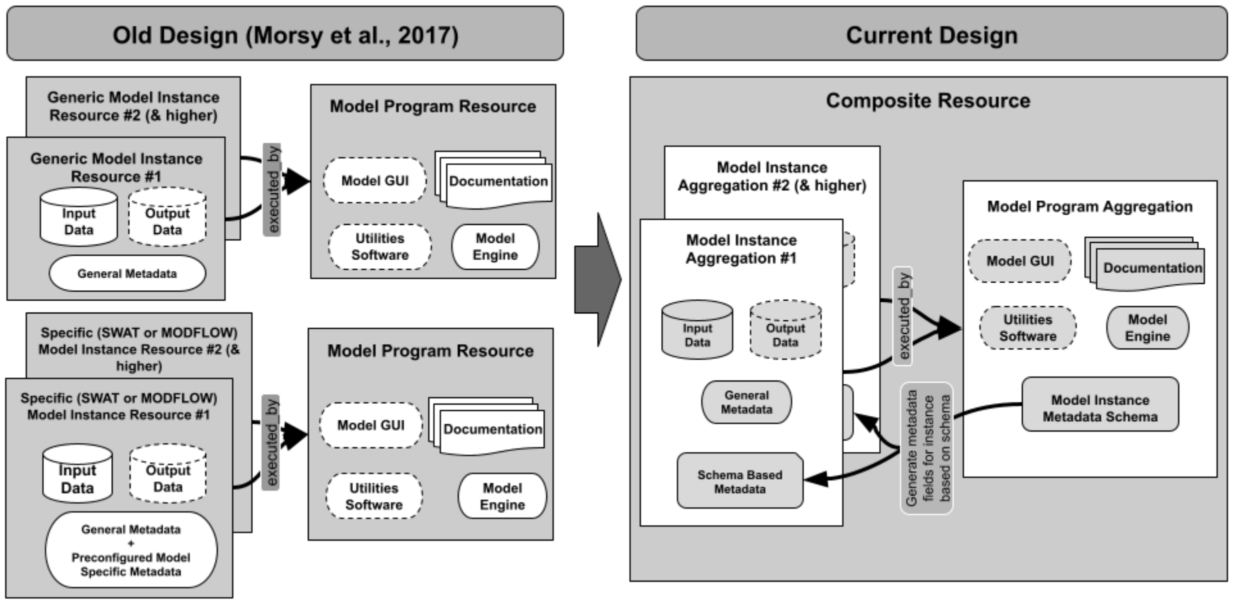

Old Design (Morsy et al., 2017)
Current Design
Generic Model Instance Resource #2 (& higher)
Generic Model Instance Resource #1
Input Data
Output Data
General Metadata
executed_by
Model Program Resource
Model GUI
Documentation
Utilities Software
Model Engine
Specific (SWAT or MODFLOW) Model Instance Resource #2 (& higher)
Specific (SWAT or MODFLOW) Model Instance Resource #1
General Metadata + Preconfigured Model Specific Metadata
Composite Resource
Model Instance Aggregation #2 (& higher)
Model Instance Aggregation #1
Model Program Aggregation
Schema Based Metadata
Generate metadata fields for instance based on schema
Model Instance Metadata Schema

Figure 2.1. Key components of the new MP and MI aggregations (current design; right) versus those of obsolete MP and MI resources (old design by Morsy et al. (2017); left)

In this study, the core of Morsy et al.'s (2017) design was kept for the same reasons mentioned by their study (Figure 1, right side). The core design includes: (1) considering a computational model as two distinct concepts (MP and MI); (2) allowing linking of many-to-one relation between MI and the MP used for executing the MIs via an "ExecutedBy" relation, and (3) using RDF-XML for storing the encoded Dublin Core metadata and extended metadata introduced in the previous and current research. However, the necessary changes required to make the design more scalable and flexible are discussed in the following paragraphs.

First and foremost, the HydroShare model resource types were converted to model content aggregation types consistent with HydroShare's composite resource design. No distinct "resources" for MI and MP exist in the new implementation. This followed the same transition that happened for all of HydroShare's typed resources. Given the difficulty in defining and implementing specific resource types in the HydroShare code, a single, "composite" resource type was defined within which content of different types could be uploaded and shared. This redesign had several benefits that included simplifying HydroShare's code, enabling content of multiple different types to be shared within a single resource and receive a single citation, and making it easier to define and implement typed content. These changes were focused on simplifying functionality while improving flexibility for storing and organizing contents in HydroShare. Indeed, we observed that too much complexity may lead to users not adopting the approach broadly. For example, having separate resource types for models and data in the older versions of HydroShare (including the versions that used Morsy et al.'s (2017) design) could be a reason for some users not to utilizing the developed features fully. When exploring the resources shared in HydroShare, we observed that some users simply shared all of their models and data within a composite resource instead of using HydroShare's specific model and data resource types. Given this and other evidence supporting HydroShare’s composite resource design, only one resource type now exists in HydroShare that can contain multiple content type aggregations, including data (i.e., multi-dimensional, geographic feature, geographic raster, time series, referenced time series, file-set, and single file) and models (MI and MP). Consequently, all existing specialized HydroShare resource types were converted to content aggregation types in the composite resource (e.g., the multi-dimensional space-time data aggregation described by Gan et al. (2020)).

HydroShare’s content type aggregations extend HydroShare’s resource-level Dublin Core metadata to provide new metadata elements that support functionality specific to common hydrologic datasets or other content types. Content type aggregations consist of one or more related content files visualized with a single icon in HydroShare's file browser and may also have content-level metadata and specialized functionality available within HydroShare’s web user interface or via HydroShare’s linked web apps. Additionally, in the previous design and implementation, specifically typed data could not be present in MI resources (e.g., multi-dimensional space-time data could not be a recognized aggregation inside a MI). In this new design and implementation, different data types can be present in a MI content aggregation with automatic metadata recognition. Metadata elements for HydroShare resources holding any content types (for example, MI and MP content types) are either resource-level or content aggregation-level metadata. Figure 2.2 shows the resource-level metadata elements common to all HydroShare resources describing a resource's general attributes. They are based on the standard Dublin Core metadata elements prefixed with “dc” (e.g., title, creator, abstract, keywords) and elements that extended “dc” prefixed with “hsterms” (e.g., sub-elements of creator: name, organization, creatorOrder, email that do not exist in the Dublin Core namespace). Content-level metadata for MI and MP are described in the following subsections.

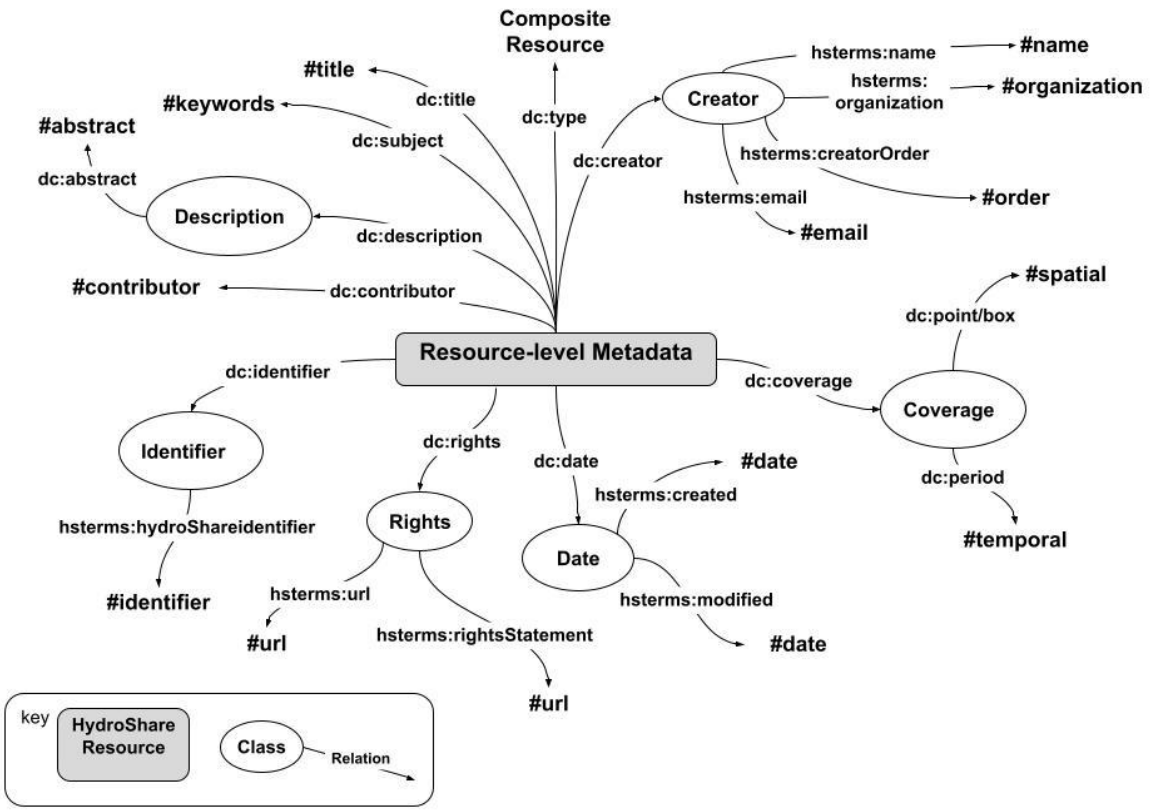


Figure 2.2. Resource-level metadata elements for a composite resource expressed as RDF triples. The # prefix signifies an attribute that can be populated when creating a composite resource. Each Dublin Core metadata element is prefixed with “dc”; each metadata element defined by HydroShare is prefixed with “hsterms”

The second change is that the four different model resource types defined by Morsy et al. (2017) (generic MP, generic MI, SWAT MI, and MODFLOW MI) were replaced by two model content aggregation types as part of the composite resource: MP content aggregation and MI content aggregation. In contrast, no specific MIs for SWAT and MOFLOW are provided in the new design (compare Figure 1, left side against right side). Morsy et al. (2017) could only support specific MIs with rich metadata for a limited number of models (i.e., SWAT and MODFLOW) with the ability to support any other type of MIs as generic MI but with little metadata description. This limitation in the design was due to the fact that such a task requires significant time, effort, and detailed knowledge of dozens of hydrological models, which is beyond the capabilities of a limited number of people (i.e., repository developers). Therefore, in the new design, we provided a more scalable design by enabling the users to support specific metadata for any arbitrary hydrological and environmental models of their choice.

Last but not least, to have a more flexible design, a machine-readable, user-owned MI metadata schema (referred to as “metadata schema” in this paper) was introduced. The MI metadata can be redefined using the metadata schema by repository users (the modelers themselves), who are most knowledgeable and invested in describing the MIs of an MP, without additional effort by HydroShare developers as opposed to Morsy et al. (2017) 's design.

The metadata for MPs and MIs and the MI metadata schema are described in more detail in the following subsections.

2.2.1.1 Model program aggregation metadata

Similar to Morsy et al. (2017), the MP content aggregation includes the software and files necessary to identify, install, and run a given environmental and hydrological model. A MP usually includes the model engine (the core mathematical modeling logic for the model; Morsy et al., (2014 and 2017)), and often, but not always, a graphical user interface (GUI) and other utility software. As an example of the case were different MPs have same or similar model engines, Morsy et al. (2017) noted there are multiple MPs that has different GUI but same or similar Storm Water Management Model (SWMM) as their model engine. We followed the basic design by Morsy et al. (2017) to link a MP with a MI instead of a model engine with an MI. It is possible that two independent MPs that use the same original model engine do not produce the same model output; therefore, we followed the old design of the MP.

Figure 2.3 shows content type metadata elements (or aggregation-level metadata) designed to describe the MP content aggregation. These metadata elements are similar to those of Morsy et al. (2017), and only for the sake of completeness, we briefly explain them here. When identifying metadata for a MP, the goal was to define a specific version of the software, its computer system compatibility, and its appropriate and intended use. To promote interoperability, this metadata consists of general elements, which are based on Dublin Core metadata elements (shown in Figure 2.3 using the “dc” prefix), to capture the basic information of any content type (e.g., title, keywords, and coverage). Then, similar to Morsy et al. (2017), the basic Dublin Core metadata elements were extended by content type developers with specific metadata (shown in Figure 3 using the “hsterms” prefix) for MP content type (e.g., modelEngine, modelVersion and modelReleaseDate). The MP content aggregation-level metadata elements can be used for several purposes, such as enhanced search and discovery across a large number of MPs. Additionally, this metadata can help improve the reproducibility of research by capturing the exact MP used to execute a particular MI.

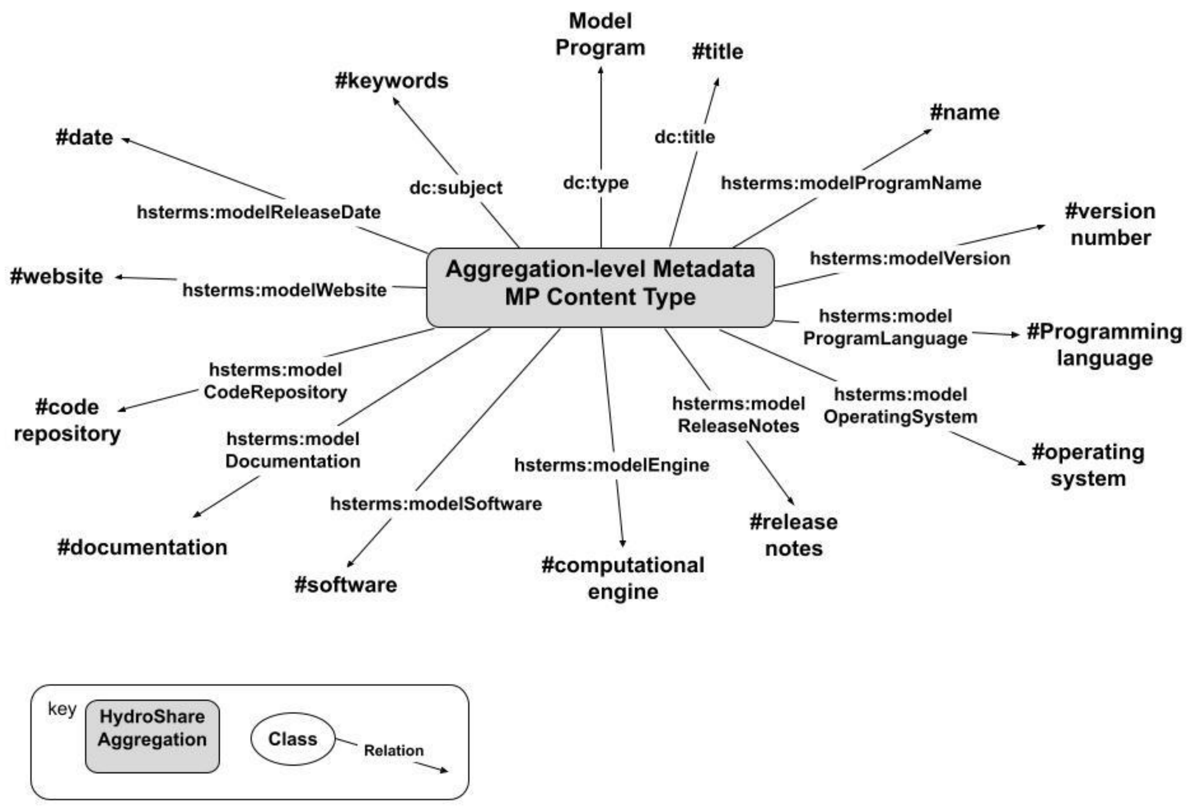


Figure 2.3. MP content aggregation metadata elements expressed as RDF triples. The # prefix signifies an attribute that can be populated when implementing the metadata for a given MP. Each Dublin Core metadata element is prefixed with “dc”; each metadata element defined by HydroShare is prefixed with “hsterms”. Modified from Morsy et al. (2017)

2.2.1.2 Model instance aggregation metadata

The MI content aggregation describes the input files required for applying a hydrological model to a specific location (Morsy et al., 2017). It may also optionally include the output files resulting from model execution. The reason for making the inclusion of output files optional was that model outputs might be massive and can be regenerated using the same MP (for example, output files generated by 2D models like HEC-RAS and TUFLOW). Many MI content aggregations can be executed by and linked to a single MP content aggregation in the same way provided by Morsy et al. (2017). When re-designing the MI metadata, our goal was to define and distinguish between different MIs across a wide range of environmental and hydrological models. Similar to Morsy et al. (2017), MI content aggregation metadata includes general elements based on Dublin Core (shown in Figure 2.4 using the “dc” prefix) to capture the basic information of any content aggregation (e.g., title, keywords, and coverage). Then, following Morsy et al. (2017), the basic Dublin Core metadata can be extended by modelers with MI content aggregation-specific metadata (shown in Figure 2.4 using the “hsterms” prefix; executedByModelProgram and includesModelOutputs metadata are from the previous design). In addition to the metadata terms first designed by Morsy et al. (2017), we now allow HydroShare users to extend the design using a MI metadata schema that defines model-specific metadata for any MI according to their needs. A MI metadata schema is a user-definable and machine-readable JSON file that extends metadata elements by generating specific, extended metadata elements to describe a MI. A wide variety of environmental and hydrologic models require different metadata to describe them, and there is no consensus on any given set of metadata that can describe a specific MI. So, using a metadata schema, modelers can decide which metadata to include according to their knowledge and needs. Design of the MI metadata schema is discussed in the following subsection.

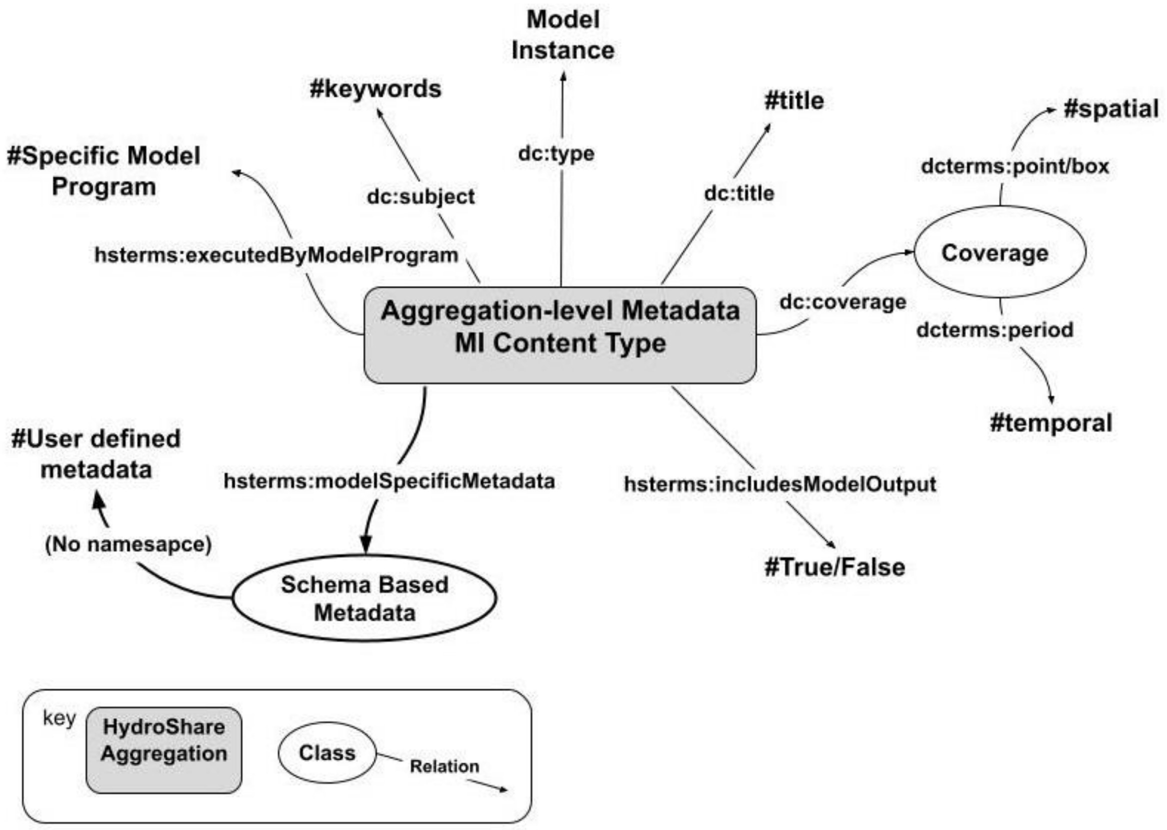


Figure 2.4. MI content aggregation metadata elements expressed as RDF triples. The # prefix signifies an attribute that can be populated when implementing the metadata for a given MI. Each Dublin Core metadata element is prefixed with “dc”; each metadata element defined by HydroShare is prefixed with “hsterms”

2.2.1.3 Model instance metadata schema

As the main aspect of redesigning the model metadata framework presented by Morsy et al. (2017), specific extended metadata for MIs of any arbitrary model now can be supported through introducing a user-definable machine-readable MI metadata schema**.** The MI metadata schema generates specific extended metadata elements for the MI (Figure 1, right side).

Using the MI metadata JSON schema, a modeler can define the metadata elements (i.e., properties such as title), semantics of the elements (i.e., descriptions of the elements and acceptable values), and rules for extended metadata element content (i.e., properties such as data type and default value; whether the element is mandatory, optional, or conditional; whether the element has nested elements inside it; whether the element's value must be chosen from a set of controlled values; and whether it is a multiple checkbox field). Therefore, the sharing and documenting of models are facilitated by giving ownership to users/modelers through leveraging the MI metadata schema, which addresses the flexibility and scalability problems that existed in the previous design.

We chose to encode the MI metadata schema using a JavaScript Object Notation (JSON) Schema document (IETF, 2013). JSON is an open, standard, lightweight text format for serializing structured data based on the data types of the JavaScript programming language. JSON encodes data objects consisting of attribute-value pairs and arrays. JSON Schema is a JSON-based format for defining the structure of JSON data. We chose JSON Schema for the MI metadata schema because it is widely recognized as fit for this purpose, can be easily understood by both machines and developers, and is self-describing. There are widely accepted standards available for the JSON Schema file format (Pezoa et al., 2016), and several packages (for Python and other programming languages) are available for parsing JSON files. HydroShare users who want to define a metadata schema for a MI within a HydroShare resource can upload a custom JSON schema file in HydroShare. To make this easier, we developed two example JSON schemas that are provided by HydroShare for the SWAT and MODFLOW models. These example JSON schema documents can be used in the following ways: 1) for SWAT and MODFLOW models with no further changes, or 2) as examples that can be modified to create a JSON schema for any other hydrologic models (including SWAT and MODFLOW) as per users' needs. In the following subsection, SWAT_schema_1.0.0 is explored as an example MI metadata schema provided by HydroShare.

2.2.1.3.1 SWAT_schema_1.0.0.json: An example model instance metadata schema

The SWAT_schema_1.0.0.json is a JSON metadata schema document that can be used to describe any SWAT MI. This SWAT MI metadata schema is provided as an example in HydroShare and was designed so that the metadata elements resemble those of the obsolete SWAT MI resource type previously introduced by Morsy et al. (2017) as close as possible. This guaranteed that the SWAT JSON schema is compatible with the SWAT MI resources created by HydroShare users using the previous design. For more details on the reasons for choosing specific metadata elements, the readers are encouraged to refer to Morsy et al. (2017). The extended metadata elements for a SWAT MI using SWAT_schema_1.0.0 are shown in Figure 2.4. Despite being specific and extensive, many of these elements are optional, which keeps the barrier to entry low. That said, we encourage users to populate as many metadata elements as possible to increase discovery and reuse of their HydroShare resources.

The SWAT MI metadata schema introduces controlled vocabularies for the SWAT model metadata elements of simulationType and simulationTimeStepType. These controlled vocabularies are compatible with the controlled vocabularies used by SWATShare, an interactive web platform used to share, run, and visualize SWAT models online (Rajib et al., 2016). The

SWATShare webapp, which is linked to HydroShare, lets users to run and visualize SWAT model instances that are stored in HydroShare. One example of the controlled vocabulary is simulationTimeStepType which has a controlled vocabulary consisting of four choices: annual, monthly, daily, and hourly. For modelObjective, the SWAT MI metadata schema allows the user to select from a list of available model objectives or define new ones.

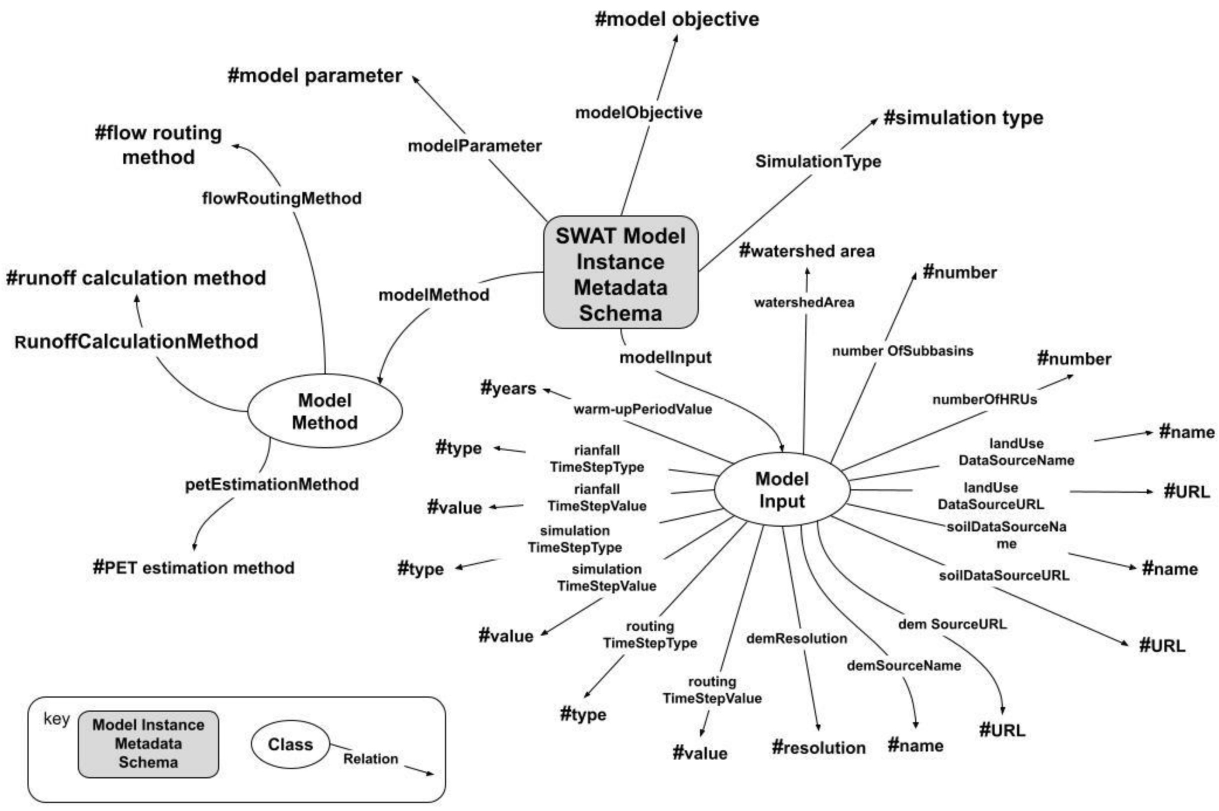


Figure 2.5. Graphical depiction of the SWAT MI metadata schema (SWAT_Schema_ 1.0.0.json) provided as an example by HydroShare. Modified from Morsy et al. (2017)

2.2.2 Experimental use case

To demonstrate our model metadata approach, we published the data used by a previous modeling study by Ercan et al. (2020) as a HydroShare resource (Maghami, 2021). Their work quantified possible changes in the water balance of a 1373 km^2 watershed in North Carolina, the Upper Neuse watershed, due to climate change. They first did a sensitivity analysis to find out the most sensitive model parameters in their study area. Next, they calibrated and validated the SWAT model using daily streamflow records within the watershed. Then, they used the SWAT model forced with different climate scenarios for baseline, mid-century, and end-century periods using five different downscaled General Circulation Models.

The fact that Ercan et al. (2020) did not formally publish the data or MIs used in their study made it a good motivation for us to publish their data and MIs as a real example that demonstrates our design's capabilities and implementation in HydroShare. The HydroShare resource we created consists of five MIs that are linked to one MP (Figure 2.6). The developed HydroShare resource also ensures that the MIs and MP used by Ercan et al. (2020) are captured and are available to other researchers. Other researchers can follow the steps we used in this use case as an established example to publish their MIs and MPs.

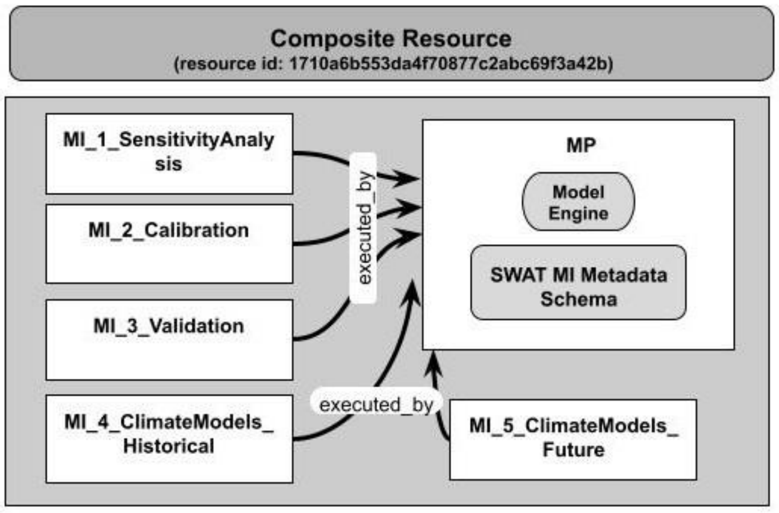


Figure 2.6. SWAT use case implementation: the resource holding five MI and one MP aggregation

2.3 Results

2.3.1 Software implementation within HydroShare

HydroShare's resource data model is based on a general design that enables new content types (aggregations) to be created by inheriting from the abstract content type. We created two new content types for MP and MI. Figure 2.7 show the UML diagram of the logical design for the MP and MI content types in HydroShare. Two categories of UML classes exist for each content types: (1) the abstract classes and (2) the specific content type classes. Abstract classes (i.e., the AbstractLogiclFile, the AbstractFileMetaData, and the AbstractModelLogicalFile) are inherited by any content type. The specific content type classes are unique to each content type and define that content type. The ModelInstanceLogicalFile class, and ModelInstanceFileMetaData class define the MI content type while the ModelProgramLogicalFile class, ModelProgramFileMetaData class, and ModelProgramResourceFileType class define the MP content type aggregation.

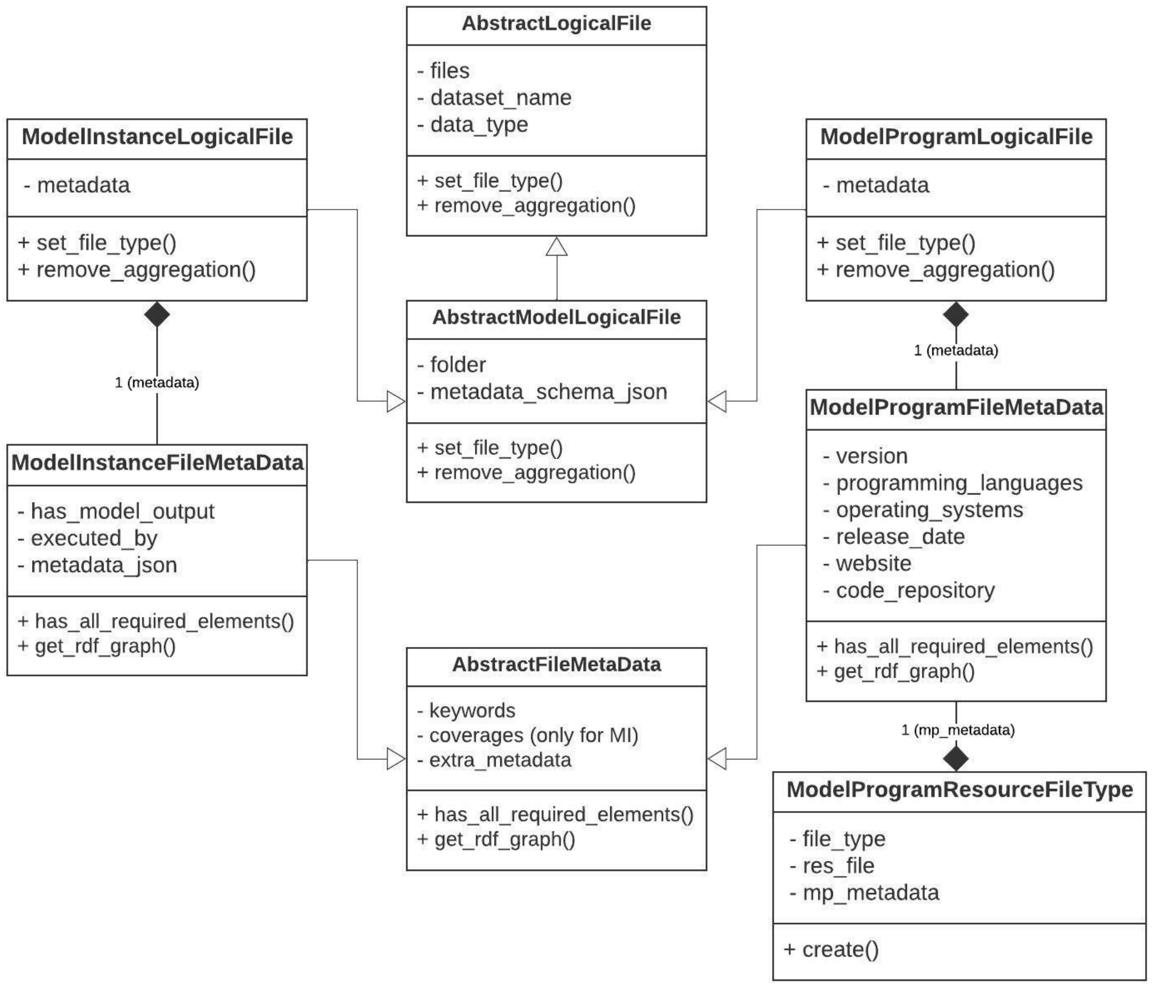


Figure 2.7: UML class diagram for the MP and MI content type implemented in HydroShare

2.3.2 Results from the example use case

Figure 2.8 shows the resource-level metadata for the HydroShare resource that contains the five MIs and one MP. This metadata provides information such as the title, abstract and other metadata following the Dublin core metadata standard (shown using “dc” prefix). Additionally, extended metadata (shown with “hsterms” prefix) helps provide a more complete description of the resource (e.g., hydroShareidentifier).

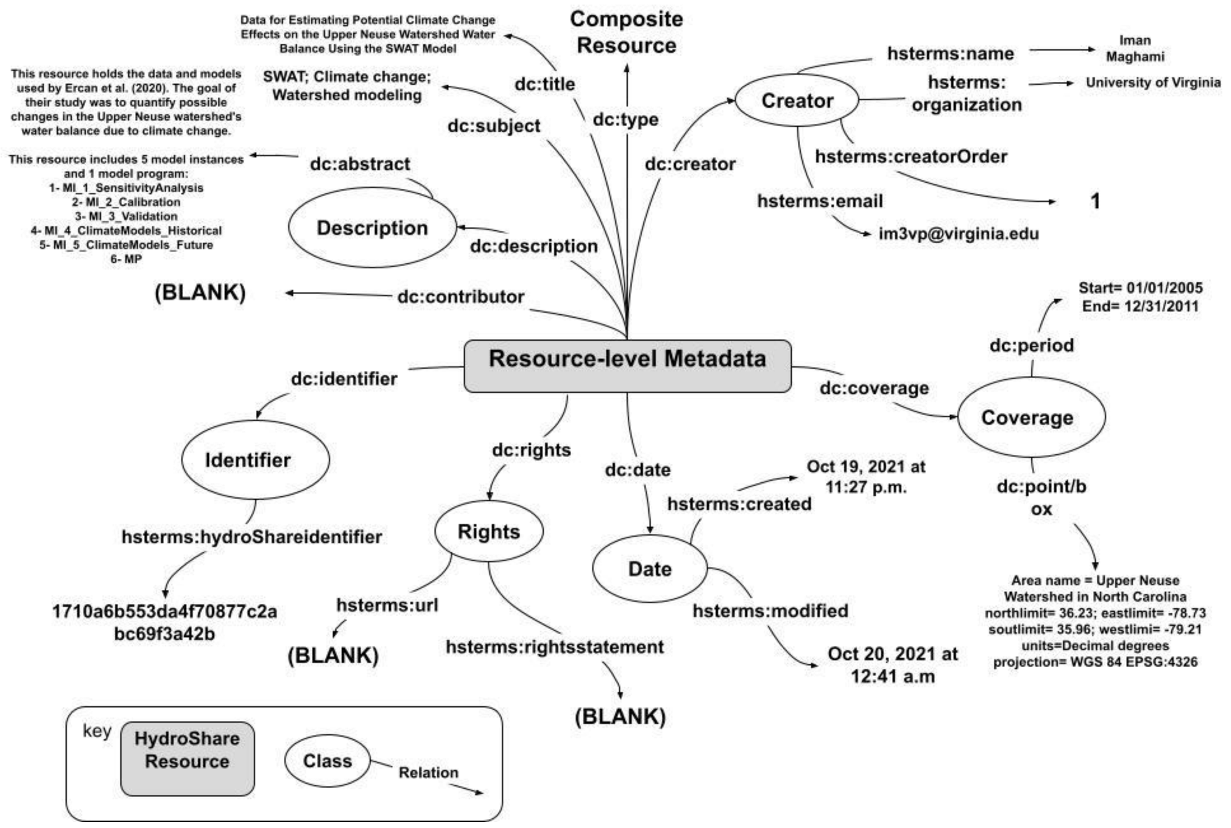

Data for Estimating Potential Climate Change Effects on the Upper Neuse Watershed Water Balance Using the SWAT Model
Composite Resource
dc:title
SWAT; Climate change; Watershed modeling
dc:subject
dc:type
This resource holds the data and models used by Ercan et al. (2020). The goal of their study was to quantify possible changes in the Upper Neuse watershed's water balance due to climate change.
This resource includes 5 model instances and 1 model program:
1- MI_1_SensitivityAnalysis
2- MI_2_Calibration
3- MI_3_Validation
4- MI_4_ClimateModels_Historical
5- MI_5_ClimateModels_Future
6- MP
dc:abstract
Description
dc:description
(BLANK)
dc:contributor
dc:creator
Creator
hsterms:name
Iman Maghami
hsterms: organization
University of Virginia
hsterms:creatorOrder
1
hsterms:email
im3vp@virginia.edu
Resource-level Metadata
dc:identifier
Identifier
hsterms:hydroShareidentifier
1710a6b553da4f70877c2a bc69f3a42b
dc:rights
Rights
hsterms:url
(BLANK)
hsterms:rightsstatement
(BLANK)
dc:date
Date
hsterms:created
Oct 19, 2021 at 11:27 p.m.
hsterms:modified
Oct 20, 2021 at 12:41 a.m
dc:coverage
Coverage
dc:period
Start= 01/01/2005
End= 12/31/2011
dc:point/b ox
Area name = Upper Neuse Watershed in North Carolina
northlimit= 36.23; eastlimit= -78.73
soutlimit= 35.96; westlimi= -79.21
units=Decimal degrees
projection= WGS 84 EPSG:4326
key
HydroShare Resource
Class
Relation

Figure 2.8. Results of populating resource-level metadata for the composite resource of the example use case

Figure 2.9 shows generic aggregation-level metadata for one of the SWAT MIs created for the case study. The title of this MI is “MI_1_SensitivityAnalysis”. In Figure 2.9, the schema based metadata for this specific MI was generated using the SWAT_Schema_1.0.0.json metadata schema file (Figure 2.5). Figure 2.10 illustrates the captured metadata for the same MI using the SWAT_Schema_1.0.0.json (Figure 2.5).

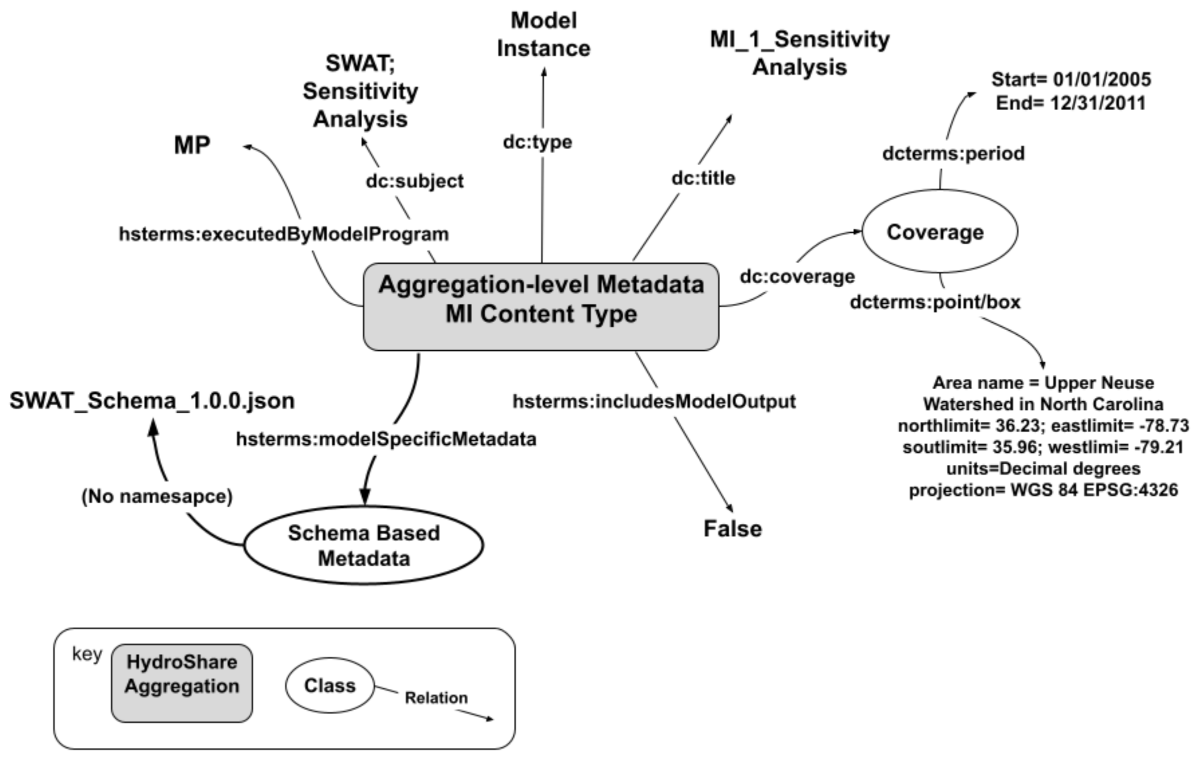

Model Instance
dc:type
SWAT; Sensitivity Analysis
dc:subject
MP
hsterms:executedByModelProgram
MI_1_Sensitivity Analysis
dc:title
Start= 01/01/2005 End= 12/31/2011
dcterms:period
Coverage
dc:coverage
dcterms:point/box
Aggregation-level Metadata MI Content Type
Area name = Upper Neuse Watershed in North Carolina
northlimit= 36.23; eastlimit= -78.73
soutlimit= 35.96; westlimi= -79.21
units=Decimal degrees
projection= WGS 84 EPSG:4326
SWAT_Schema_1.0.0.json
hsterms:modelSpecificMetadata
hsterms:includesModelOutput
(No namesapce)
Schema Based Metadata
False
key
HydroShare Aggregation
Class
Relation

Figure 2.9. Results of populating aggregation-level metadata for one of the MIs of the example use case

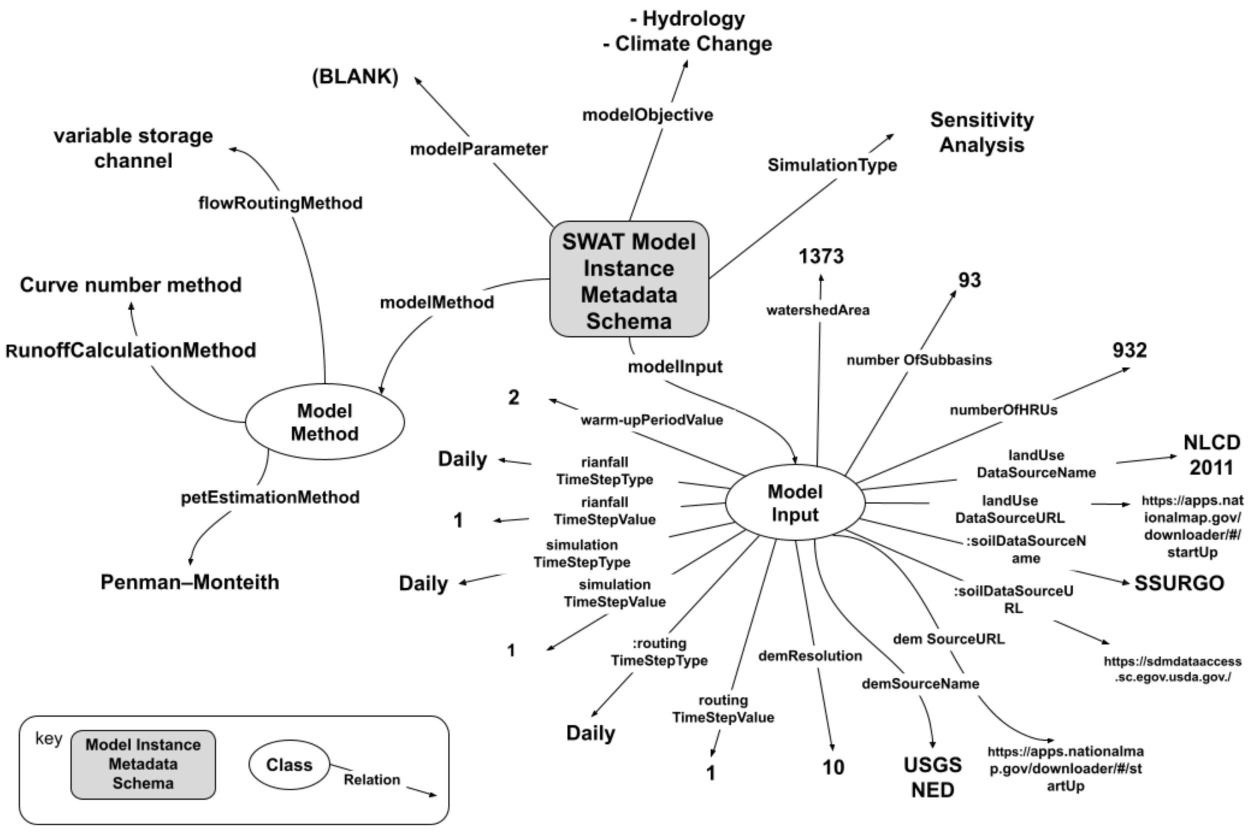


Figure 2.10. Results of populating the schema-based metadata using the SWAT MI metadata schema for the example use case (SWAT_Schema_1.0.0.json)

Figure 2.11 shows an activity diagram demonstrating the steps necessary for creating MP and MI content aggregations in a resource within the HydroShare repository. As stated before, in this example, one composite resource containing five MP aggregations and one MI content aggregation were created.

Once the content of the resource is complete, the sharing status of the resource can be changed to discoverable or public. If the user chooses to permanently publish the resource, a more formal digital object identifier (DOI) would be assigned to the resource. When resources are permanently published, the users can no longer edit the content files or the author and title metadata elements. In contrast, users can still edit some of the less fundamental metadata elements (e.g., the content-level metadata for a MI content aggregation).

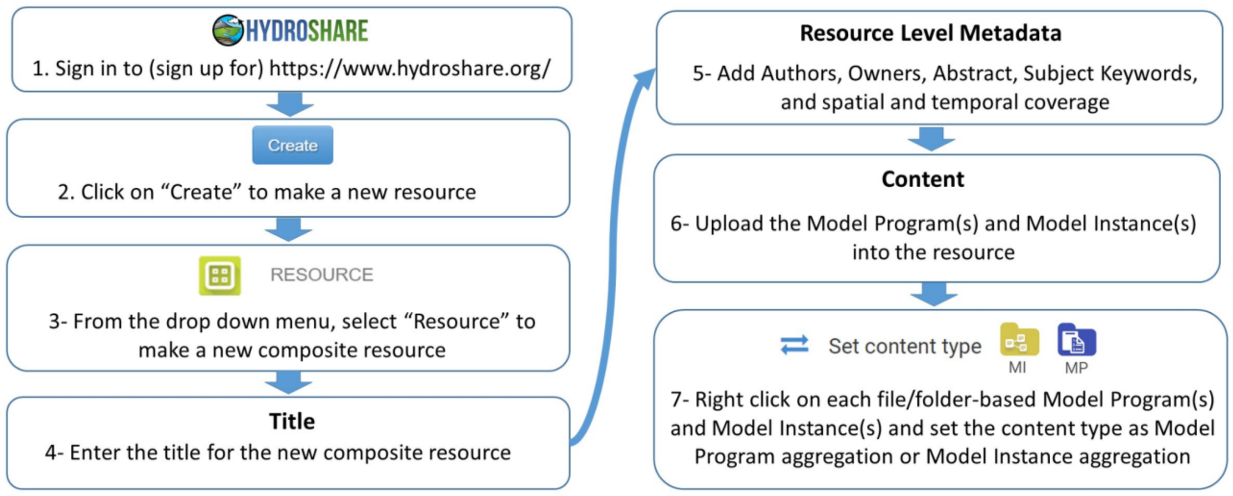
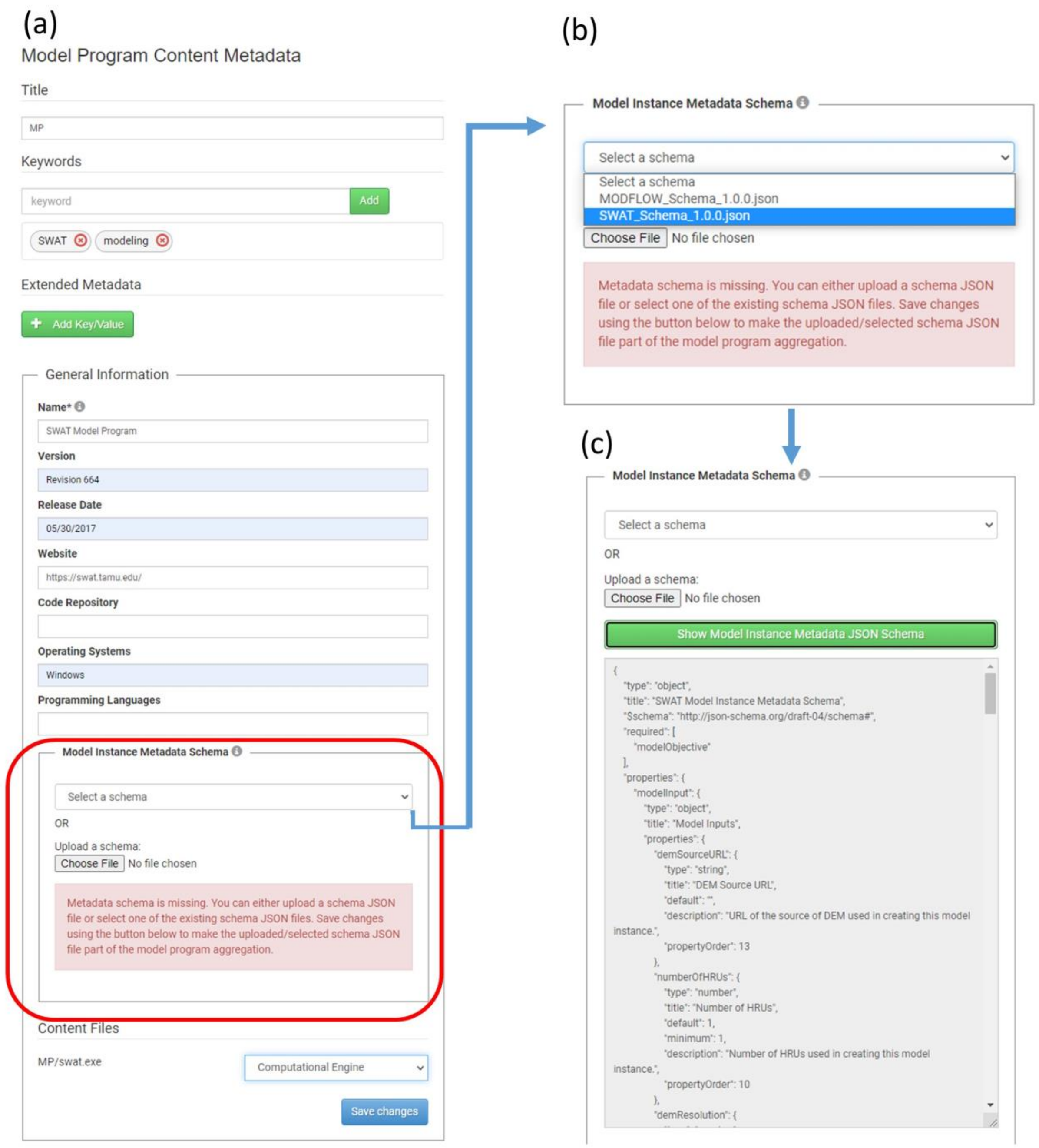

HYDROSHARE
1. Sign in to (sign up for) https://www.hydroshare.org/
Create
2. Click on "Create" to make a new resource
RESOURCE
3- From the drop down menu, select "Resource" to make a new composite resource
Title
4- Enter the title for the new composite resource
Resource Level Metadata
5- Add Authors, Owners, Abstract, Subject Keywords, and spatial and temporal coverage
Content
6- Upload the Model Program(s) and Model Instance(s) into the resource
Set content type
MI
MP
7- Right click on each file/folder-based Model Program(s) and Model Instance(s) and set the content type as Model Program aggregation or Model Instance aggregation

Figure 2.11. Activity diagram showing the steps to create MP and MI within HydroShare

2.3.2.1 Creating the MP aggregations and selecting a MI metadata schema

In this example use case, we created one folder for the MP. This folder is called "MP" and includes a single content file called "SWAT.exe". This file is the executable used by Ercan et al. (2020) to execute the SWAT model for our example use case. After uploading the file, the folder was converted to a MP content aggregation by right-clicking on it and selecting "Set content type → Model Program". Figure 2.12 shows the MP content-related metadata in the panel on the right side of the HydroShare file browser interface where it can be edited. It shows the standard set of metadata for a MP aggregation that includes general information (title, Keywords) and MP specific metadata (e.g., modelProgramName, modelVersion) for our use case on the HydroShare website. Figure 2.12 b shows that users can either select a metadata schema from a dropdown menu (currently only MODFLOW and SWAT example schemas are provided) or upload a pre-defined JSON metadata schema file. The selected metadata schema will be used to define the metadata requirements of any MI content aggregations linked to this MP using the "ExecutedBy" metadata field of the MI. Figure 2.12 c shows the content of the metadata schema in JSON file format as displayed in HydroShare.



Model Program Content Metadata

Title: MP

Keywords: SWAT, modeling

Extended Metadata

General Information

Name*: SWAT Model Program
Version: Revision 664
Release Date: 05/30/2017
Website: https://swat.tamu.edu/
Code Repository:
Operating Systems: Windows
Programming Languages:

Model Instance Metadata Schema

Select a schema

OR

Upload a schema: No file chosen

Metadata schema is missing. You can either upload a schema JSON file or select one of the existing schema JSON files. Save changes using the button below to make the uploaded/selected schema JSON file part of the model program aggregation.

Content Files

MP/swat.exe — Computational Engine

Model Instance Metadata Schema

Select a schema
MODFLOW_Schema_1.0.0.json
SWAT_Schema_1.0.0.json

No file chosen

Metadata schema is missing. You can either upload a schema JSON file or select one of the existing schema JSON files. Save changes using the button below to make the uploaded/selected schema JSON file part of the model program aggregation.

Model Instance Metadata Schema

Select a schema

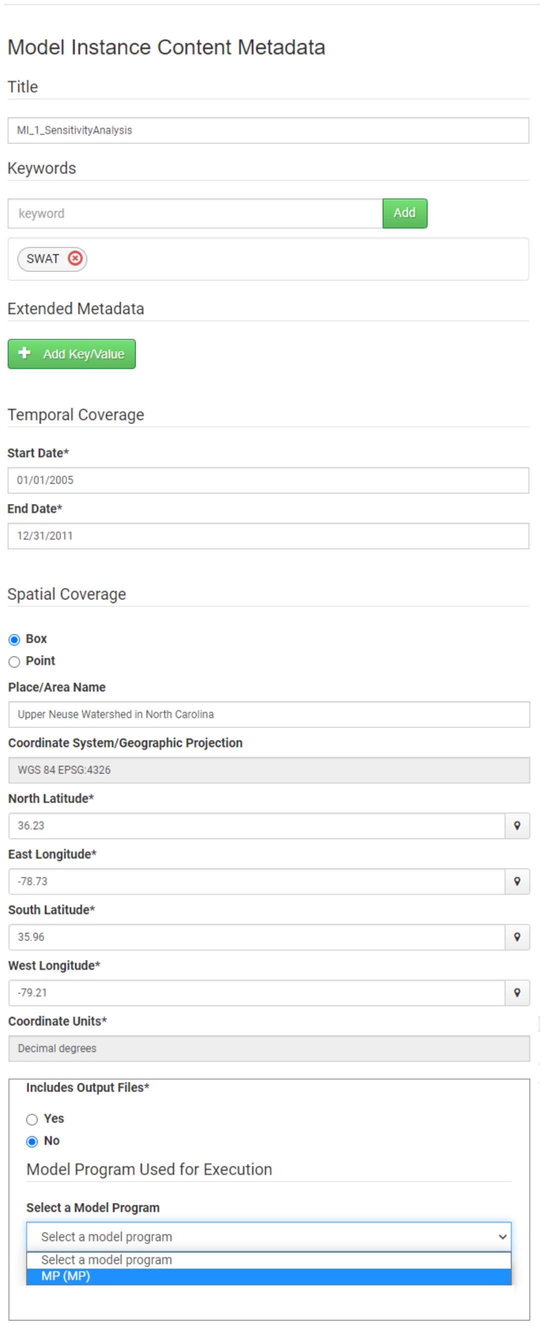
OR

Upload a schema: No file chosen

```
{
   "type": "object",
   "title": "SWAT Model Instance Metadata Schema",
   "$schema": "http://json-schema.org/draft-04/schema#",
   "required": [
      "modelObjective"
   ],
   "properties": {
      "modelInput": {
         "type": "object",
         "title": "Model Inputs",
         "properties": {
            "demSourceURL": {
               "type": "string",
               "title": "DEM Source URL",
               "default": "",
               "description": "URL of the source of DEM used in creating this model
instance.",
               "propertyOrder": 13
            },
            "numberOfHRUs": {
               "type": "number",
               "title": "Number of HRUs",
               "default": 1,
               "minimum": 1,
               "description": "Number of HRUs used in creating this model
instance.",
               "propertyOrder": 10
            },
            "demResolution": {
```

Figure 2.12. MP metadata and steps to select/upload MI Metadata Schema on a resource landing page within HydroShare (shown in edit mode)

Within the same HydroShare resource, we created a folder called "MI_1_SensitivityAnalysis", which contains the content files for the first MI of the example use case. Then, the folder was converted to a MI content aggregation by right-clicking on it and selecting "Set content type → Model Instance". Figure 2.13 shows the MI content aggregation-related metadata, which is shown in the panel on the right side of HydroShare's file browser interface. It shows the standard set of metadata that includes general information (title, Keywords) and MI-specific metadata (includesmodelOutput and executedByModelProgram) for our use case on the HydroShare website.



Model Instance Content Metadata

Title

MI_1_SensitivityAnalysis

Keywords

keyword Add

SWAT

Extended Metadata

Add Key/Value

Temporal Coverage

Start Date*: 01/01/2005

End Date*: 12/31/2011

Spatial Coverage

[✓ Box] Point

Place/Area Name: Upper Neuse Watershed in North Carolina

Coordinate System/Geographic Projection: WGS 84 EPSG:4326

North Latitude*: 36.23

East Longitude*: -78.73

South Latitude*: 35.96

West Longitude*: -79.21

Coordinate Units*: Decimal degrees

Includes Output Files*

Yes [✓ No]

Model Program Used for Execution

Select a Model Program: Select a model program

Select a model program

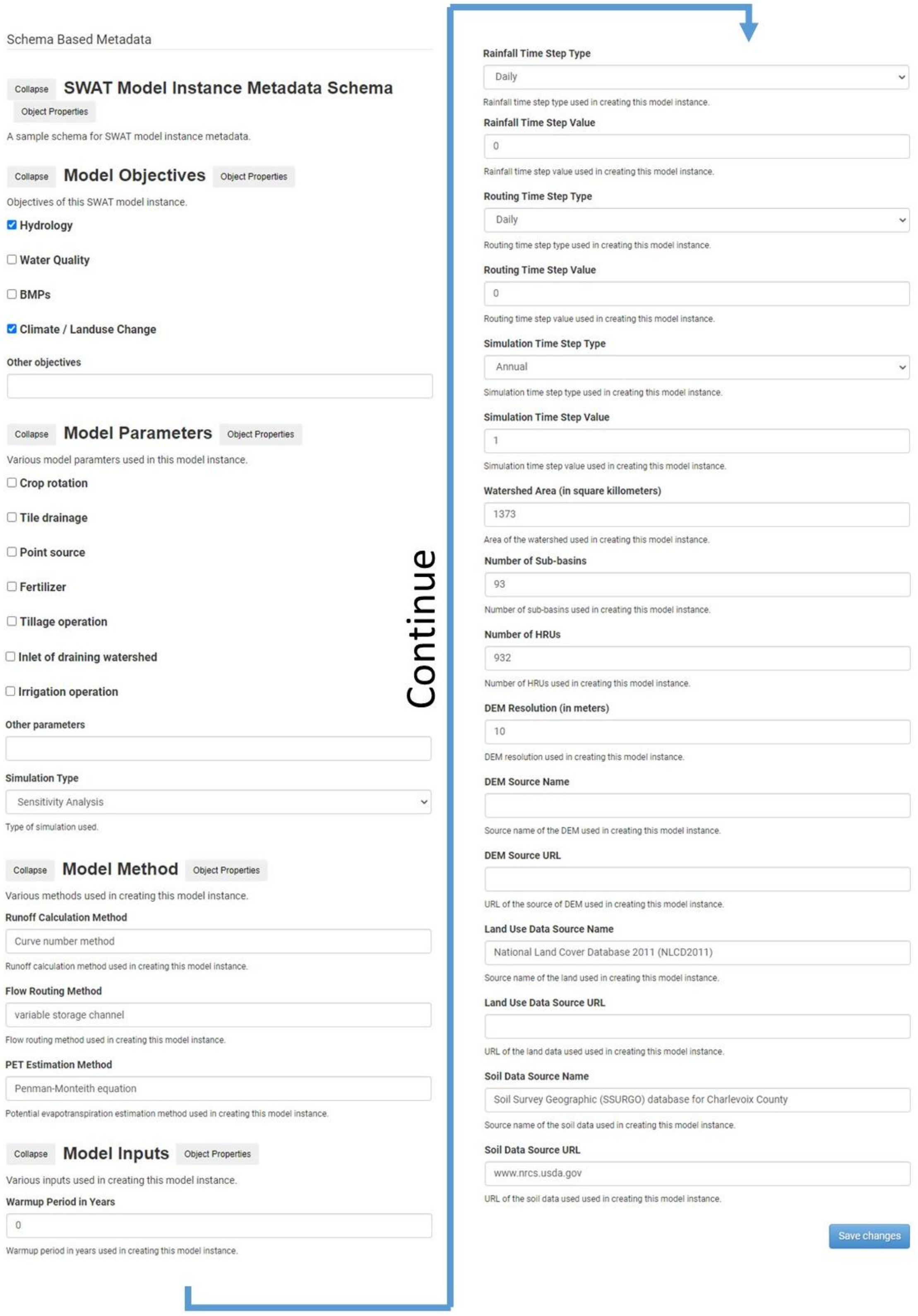
MP (MP)

Figure 2.13 MI standard and content-specific metadata on a resource landing page within HydroShare (shown in edit mode)

Once the user links the MI content aggregation to the MP content aggretation to which the JSON metadata schema has been added, they can see and edit the model-specific metadata (schema-based metadata) for the MI. The JSON schema file that specifies the MI metadata is used to dynamically build the web form for displaying and editing the MI metadata (Figure 2.14). The same JSON schema file is used to validate metadata entered by the user (e.g., required and optional elements, data types, etc.).

2.3.2.2 Updating the MI metadata schema

After schema-based metadata has been added to a MI, if a user changes the JSON metadata schema used for the linked MP, the existing MI schema-based metadata may or may not be valid against the new schema. Suppose the schema-based metadata is still valid, after switching to edit mode in the MI. In that case, the user will notice that a blue button appears, enabling the user to "Update Metadata Schema from Model Program" (Figure 2.15 a). The blue button copies the updated metadata schema from the MP over to the MI safely, and no loss of metadata occurs. One example of where this may occur is when a user edits an existing JSON metadata schema by adding a new attribute that is NOT set to be required. However, if the linked MP has a schema that invalidates the existing metadata in a MI, the metadata panel for MI content aggregation will look like Figure 2.15 b (with the option to update/copy schema showing as a red button). The red button to copy the metadata schema from the MP over to the MI will perform a non-safe update – it will result in loss of metadata in the MI (as indicated by the message that appears above the update button). One example of this case is when a user edits an existing JSON schema by removing an attribute that already has a value in the MI's metadata.



Schema Based Metadata

Collapse **SWAT Model Instance Metadata Schema** Object Properties

A sample schema for SWAT model instance metadata.

Collapse **Model Objectives** Object Properties

Objectives of this SWAT model instance.

- [x] Hydrology
- [ ] Water Quality
- [ ] BMPs
- [x] Climate / Landuse Change

Other objectives

Collapse **Model Parameters** Object Properties

Various model paramters used in this model instance.

- [ ] Crop rotation
- [ ] Tile drainage
- [ ] Point source
- [ ] Fertilizer
- [ ] Tillage operation
- [ ] Inlet of draining watershed
- [ ] Irrigation operation

Other parameters

Simulation Type: Sensitivity Analysis

Type of simulation used.

Collapse **Model Method** Object Properties

Various methods used in creating this model instance.

Runoff Calculation Method: Curve number method

Runoff calculation method used in creating this model instance.

Flow Routing Method: variable storage channel

Flow routing method used in creating this model instance.

PET Estimation Method: Penman-Monteith equation

Potential evapotranspiration estimation method used in creating this model instance.

Collapse **Model Inputs** Object Properties

Various inputs used in creating this model instance.

Warmup Period in Years: 0

Warmup period in years used in creating this model instance.

Rainfall Time Step Type: Daily

Rainfall time step type used in creating this model instance.

Rainfall Time Step Value: 0

Rainfall time step value used in creating this model instance.

Routing Time Step Type: Daily

Routing time step type used in creating this model instance.

Routing Time Step Value: 0

Routing time step value used in creating this model instance.

Simulation Time Step Type: Annual

Simulation time step type used in creating this model instance.

Simulation Time Step Value: 1

Simulation time step value used in creating this model instance.

Watershed Area (in square kilometers): 1373

Area of the watershed used in creating this model instance.

Number of Sub-basins: 93

Number of sub-basins used in creating this model instance.

Number of HRUs: 932

Number of HRUs used in creating this model instance.

DEM Resolution (in meters): 10

DEM resolution used in creating this model instance.

DEM Source Name

Source name of the DEM used in creating this model instance.

DEM Source URL

URL of the source of DEM used in creating this model instance.

Land Use Data Source Name: National Land Cover Database 2011 (NLCD2011)

Source name of the land used in creating this model instance.

Land Use Data Source URL

URL of the land data used used in creating this model instance.

Soil Data Source Name: Soil Survey Geographic (SSURGO) database for Charlevoix County

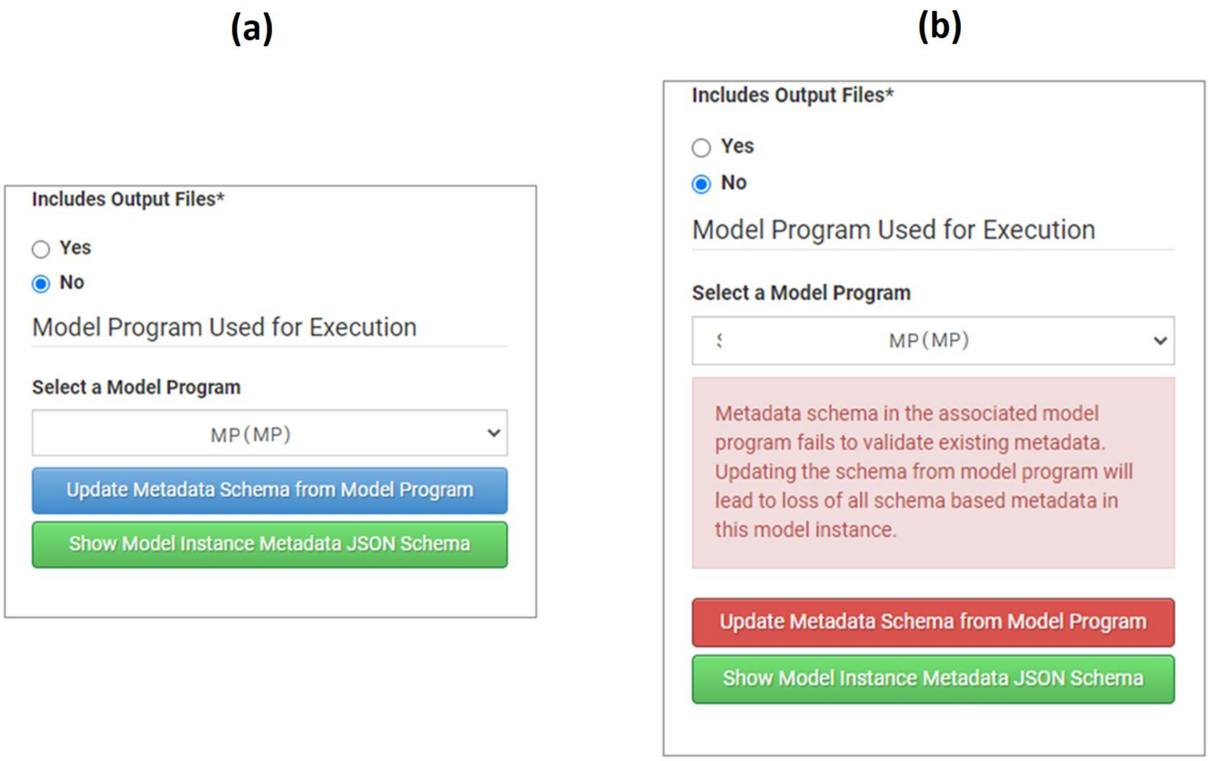
Source name of the soil data used in creating this model instance.

Soil Data Source URL: www.nrcs.usda.gov

URL of the soil data used used in creating this model instance.

Save changes

Figure 2.14. MI schema-based metadata on a resource landing page within HydroShare (shown in edit mode)


(a)
(b)
Update Metadata Schema from Model Program
Show Model Instance Metadata JSON Schema

(a)

Includes Output Files*

Yes [✓ No]

Model Program Used for Execution

Select a Model Program: MP(MP)

(b)

Includes Output Files*

Yes [✓ No]

Model Program Used for Execution

Select a Model Program: MP(MP)

Metadata schema in the associated model program fails to validate existing metadata. Updating the schema from model program will lead to loss of all schema based metadata in this model instance.

Figure 2.15. MI Schema-based metadata (a) validated, (b) NOT validated by the metadata schema in the linked MP on a resource landing page within HydroShare (shown in edit mode)

2.4 Discussion

2.4.1 Key design choices

The model metadata approach presented in this study was designed to build on and enhance the method presented by Morsy et al. (2017) to make it more scalable and flexible. To make that possible, a few critical design choices for our framework were made. Similar to Morsy et al. (2017), a MI content aggregation is allowed to be linked to only one MP content aggregation at a time. It is true that a MI may be executed by multiple MPs (e.g., a case where two MPs are of different versions but still compatible with the MI). However, using two different MPs that are compatible with the MI but have different model engines might lead to different results for a study. Therefore, for the sake of reproducibility, this design choice was made to ensure that a specific MI execution could be repeated.

Another critical design choice was that the MI (s) that are linked to a MP must be located in the same HydroShare resource as the MP. This design choice was made to ensure that the link between the MI and MP never gets corrupted. In the absence of such enforcement, where the MI content aggregation and the associated MP content aggregation were stored in different composite resources, it would be possible for the MP or the MI to be independently deleted or for the access control to one of the resources to be changed – either of which would corrupt the relationship between the MP and MI. Additionally, embedding the MI and MP in the same HydroShare resource makes it convenient to download the MI and the linked MP from HydroShare with one single action (bag download) in the same zipped folder. This may make execution of a MI easier in some instances where file specific file paths need to be preserved.

There is no limit to the number of instances of MI and MP aggregations that can exist in a single HydroShare resource. Allowing more than one MI content aggregation in a HydroShare resource allows the user to link multiple MI content aggregations to the same specific MP aggregation (as was done in our case study demonstrating a prevalent use case for users). The number of MP aggregations that can be stored in one HydroShare resource is also not restricted so that if the users need, they can put more than one MP in the resource.

Another critical design decision was to allow MI/MP aggregations to be based on a folder in addition to a file or files. This design enables MI/MP content aggregations to contain sub-folders that are beneficial in organizing the files included in a MP/MI content aggregation as this has widespread use, especially in the case of MI content aggregations that may require many different input files or types of input data.

To keep the design simple and avoid unwanted complexity, unlike Morsy et al.'s (2017) design, no nesting of MP aggregations within a MI content aggregation is allowed. We understand that there might be cases where some codes are used to process raw data and prepare the input data for a model, and users may want to embed that code as a MP aggregation within a MI content aggregation. However, for our design, MP only refers to established hydrological and environmental models (e.g., SWAT) rather than any custom preprocessing tools (e.g., codes to process the raw input data). Therefore, for the example case given, the entire raw input data, any preprocessing tools, and the processed input data are treated as a single MI. It is important to note that folder-based MI content aggregations can still contain most other HydroShare content aggregation types (e.g., geographic feature, geographic raster, multidimensional, time series, referenced time series), excluding the MP, MI or fileset type aggregations. No other aggregations (no data aggregation, MP, or MI) can be created inside a folder based MP aggregation.

Another critical design decision was that after a MI content aggregation is linked to a MP aggregation via an "ExecutedBy" relationship, if the MP content aggregation has an associated JSON metadata schema, the JSON schema file from the MP content aggregation is copied into the MI content aggregation. This is to ensure that the MI will not lose access to the JSON metadata schema if the MP were deleted. Moreover, by copying the JSON schema file into the MI content aggregation, we create a snapshot of the JSON schema file associated with the MP when the MI gets associated with the MP, and the metadata for the MI will be created using this snapshot. The editing of the JSON schema file is constrained at the MP level. Therefore, the JSON schema copy stored in the MI content aggregation can only get updated when the JSON schema of the associated MP gets updated or the MI content aggregation is linked to a new MP aggregation.

2.4.2 Migrating legacy HydroShare model resources to the new model aggregation design

It was necessary to migrate all of the legacy model resources (the resources created by HydroShare users prior to implementing the new design for model metadata described here) to model aggregations in a composite resource so that the legacy model resource types could be deprecated.

This was done so that HydroShare does not have to provide unnecessary support for obsolete model resources and to avoid the embedded complexities for code compatibility reasons, and second, to encourage modelers to use the new design. When migrating the legacy model resources to model content aggregations, we encountered some challenges, the most important of which are briefly described below.

In the new design, the MP that executes a MI needs to be in the same HydroShare resource as the MI content aggregation. However, to avoid as many changes as possible in the previously created resources, we decided not to enforce this rule for them and only keep this rule for the resources being created after the redesign. Instead, when a MI resource was linked to a MP resource, a link to the original MP resource was considered as a new metadata element for the MI content aggregation. Therefore, the following three migration scenarios were considered depending on the existence of a linkage between a MP resource and a (generic or specific) MI resource, regardless of their publication status. First, a MP resource, regardless of being linked by a MI resource, was converted to MP content aggregation within a composite resource. The resource-level metadata were kept at the resource-level for the migrated resource (i.e., title, resource id, abstract, authors, and owners) except for those that made sense to be moved to the MP content aggregation level (i.e., keywords, MP specific metadata such as version). Second, if a MI resource was not linked to a MP, it was converted to a MI content aggregation within a composite resource, and the resource-level metadata followed a similar rule as the previous scenario. Third, if a MI resource was linked to a MP, the MI resource was converted to a MI content aggregation within a composite resource, the resource-level metadata followed a similar rule as the previous scenarios, and the link to the original MP resource was considered as a new metadata element for the MI content aggregation Apart from the four scenarios, we had to store all model-specific (SWAT/MODFLOW) metadata as JSON data in the newly created MI content aggregation.

The MP and (generic or specific) MI resource being migrated, regardless of how many content files they had or whether they had a folder containing content files at the root level, were converted to composite resources containing the MP/MI content aggregation created based on a folder. That means the converted composite resource had a folder (which contains all the resource files) even though the original resource did not have a folder. The corresponding folders for MP and MI were named model-program and model-instance, respectively. If the resource being migrated had only one folder at the root level and no files at that level, the MP/MI was created based on that folder. If the resource being migrated had more than one folder or file at the resource root level, a new folder was created as part of the migration, and the existing folders/files were put under this new folder. The MP/MI content aggregation was based on the new folder.

One challenge was that we had to make sure that in the new design of model aggregation, the MI content aggregations that were for SWAT are compatible with SWATShare. Unlike Morsy et al. (2017), no specific SWAT MI is provided in the new design; therefore, we created an optional metadata element for the MI where the user could specify which web app can be used for that MI. Then, later, that metadata information can be used to display an option to “open with” a linked web application. This enables SWAT MIs to remain compatible with SWATShare.

2.4.3 Limitations of the new design and its implementation

Although the redesign of a metadata framework for MI and MP offers several important improvements, our design still has some limitations. Despite adding more flexibility for users to define extended metadata schema that better describe MPs and MIs in HydroShare, one drawback of allowing custom metadata elements defined via a JSON schema is that it makes the indexing needed for search and discovery capabilities more challenging because the metadata scope is much broader compared to the case when metadata are pre-defined, hard-coded and defined by a central group of developers. While it is difficult or impossible to anticipate all of the metadata elements that might be defined by modelers working on disparate models, the new design still uses standard Dublin Core elements useful for the discovery use case. We anticipate that patterns and commonalities in model metadata schemas will emerge through use and that these commonalities may eventually enable more advanced discovery functionality for MPs and MIs.

Another current limitation of our work, which is related to the implementation rather than the design itself, is that HydroShare does not currently allow using MI content aggregation metadata introduced by this new work for discovering resources. Enabling such discovery capability in a future release of HydroShare would be of great value. For example, it would be helpful if HydroShare could find all the MIs linked to a specific metadata schema (e.g., SWAT_schema_1.0.0).

2.4.4 Opportunities for future research

Future research to improve model content aggregation metadata should focus on establishing guidelines for designing a metadata schema from scratch – e.g., for hydrologic and environmental models that do not already have any metadata schema. This will further lower the barrier for modelers when identifying the metadata schema for any arbitrary model. Future work may also try to refine the developed model aggregation metadata and its implementation within HydroShare. While users can upload their metadata schema to HydroShare as part of the creation of a MP, currently there is no mechanism to expose the developed metadata schemas in a systematic way and use metadata to describe them. Therefore, we are considering making a database where users can upload a metadata schema by themselves. This database may track metadata on the MI metadata schema itself, such as title, creator, date created and modified, etc. to help describe the MI metadata schema so that it can be reused more easily. Currently, if users need to use a MI metadata schema developed by others, they first have to download it and then manually upload it into their MP. The database would facilitate users' access to metadata schema developed by others, and users will be able to directly associate their MI to any MI metadata schema present in the database.

We are also considering adding the ability to let the user directly copy a MP aggregation that the user has access to with one single button into their composite resource for convenience. Additionally, in the current HydroShare deployment, if updating the metadata schema leads to metadata elements not validating against the updated schema, all the populated metadata is lost, and the user needs to start over. In future releases, we can make a file difference so that the metadata elements that are still valid can remain, and the user will only need to deal with the invalid metadata. Finally, in future work we plan to add functionality such that a user can upload a code to extract the extended metadata elements automatically from MI configuration files. This will further lower the barrier for metadata entry through automatically extracting metadata that can be read from the input files. The user will then only need to populate metadata elements that are not extractable from input files.

Finally, although data and models can be shared and described using flexible and scalable metadata schemas to enhance reproducible modeling, other factors need to be taken into account. Full reproducibility requires three elements: (1) digital documenting and sharing of data and model, (2) encapsulating computational environments, and (3) encapsulating workflows (Choi et al., 2021). By sharing data and model using approaches like the one described in this paper along with encapsulated computational environments and workflows, modelers should expect to approach reproducible modeling.

2.5 Conclusions

This study presented flexible and scalable methods for describing hydrologic and environmental models with metadata. The critical redesign choice was that we introduced a standardized way of encoding user-defined metadata schemas for MIs using machine-readable JSON schema. This approach gives the capability of providing and maintaining metadata schemas for any arbitrary MP to the modeling community instead of the web-based repository developers. JSON schema files can be consumed and utilized in a scalable way, because it provides a standardized way of encoding user-defined metadata schemas and is machine-readable. This paper described the metadata schema specification file format, demonstrated its use for a SWAT modeling use case as an example, and showed how the web-based repository, HydroShare, could support and leverage this new approach for community-contributed metadata schemas for hydrologic models. In this way, support can be provided for an indefinite variety of environmental and hydrologic models with little cost to the repository developers. Extensible model metadata framework is also expected to advance the reproducibility of model-related research, an intricate issue across all science disciplines, through facilitating the sharing, discovering, and reusing of data and models built by others.

Acknowledgments

This work was supported by the National Science Foundation under collaborative grant 1664061. We greatly thank and acknowledge the work of the larger HydroShare development team.

Software Availability

The software created in this research is free and open source as part of the larger HydroShare software repository. The HydroShare repository and the model metadata functionality described in this paper are operational at https://www.hydroshare.org. The HydroShare software source code repository is managed through GitHub and is available at https://github.com/hydroshare/hydroshare.

This chapter is in preparation for submission to a peer-reviewed journal. Current coauthors of the study are Iman Maghami, Jonathan L. Goodall, Mohamed M. Morsy, Jeffrey M. Sadler, Jeffery S. Horsburgh, Pabitra K. Dash, and David G. Tarboton

Chapter 3

A Metadata Schema Design for Instances of Hydrologic Models: SUMMA as a Case Study

3.1 Introduction

The need for research reproducibility has been argued by numerous recent papers across many scientific disciplines (e.g., Nüst et al., 2018; Piccolo and Frampton, 2016; Stodden et al., 2013). In response to this need, researchers have begun to propose cyberinfrastructure designs and practices to improve computational reproducibility (e.g., Choi et al., 2021g; Essawy et al., 2018; Stodden et al., 2015). There are a broad range of computational models to address a variety of challenges in different science disciplines. These models rely on extremely heterogeneous input data, and adopt a significantly wide spectrum of structure and semantics to store model input and output data. Metadata, defined as “structured data about data” (Duval et al., 2002), is a crucial component of inventorying these model objects within an information system by helping people and machines share, discover, access, and reuse information (Essawy et al., 2020; Greenberg et al., 2013; Leipzig et al., 2021; Qin et al., 2016).

While it is acknowledged that metadata is a critical component of an information system and, by extension, computational reproducibility, it is a nearly impossible task to design a model metadata framework that can accommodate all the diverse needs of every model across all the science disciplines. Researchers from different scientific disciplines could benefit from designing flexible model metadata frameworks that are applicable to their specific science discipline and can accommodate the diverse nature of models within their discipline (Morsy et al., 2017). Reviewing the flexible metadata frameworks developed in other science disciplines, researchers can find the common concepts and design their own metadata frameworks based on their applications. In the field of environmental modeling, there have been efforts towards adopting metadata frameworks for models. Some examples are Hill et al. (2001), Wosniok and Lehfeldt. (2013), the Community Surface Dynamics Modeling System (CSDSM) (http://csdms.colorado.edu/), and OntoSoft (Gil et al., 2016, 2015). Morsy et al. (2017) introduced a generic metadata framework applicable to any hydrological and environmental models that extended the Dublin Core metadata framework (“Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1.,” 2012).

Morsy et al. (2017) defined computational model components as two distinct concepts, i.e., model program (referred to as “MP” in this paper) and model instance (referred to as “MI” in this paper). An MP resource was defined as software developed for running a model simulation and generating outputs. In contrast, an MI resource was defined as the application of the MP to a specific spatial and temporal domain. Additionally, they added an extension to their generic model metadata framework that could use specific metadata schemas to support two specific model programs and their accompanying instances for two of the most used hydrological models: Soil and Water Assessment Tool (SWAT) and Modular Three-Dimensional Finite-Difference Groundwater Flow Model (MODFLOW). "Metadata schema" as per ISO 23081, and used in the same way in their research and current research, is *"a logical plan showing the relationships between metadata elements, normally through establishing rules for the use and management of metadata specifically as regards the semantics, the syntax and the optionality (obligation level) of values"* ("The International Organization for Standardization; ISO 23081.1—s3 Terms and Definitions," 2017). Users could describe the SWAT and MODFLOW MIs with better details using the pre-defined metadata schema specific to these two models.

In Chapter 2 we argued that a flexible model metadata framework should readlily support any models, and not only a limited number of them, with the specific metadata. A bottleneck in reaching this goal was that repository developers were required to create and maintain individual metadata schemas manually while it is beyond their capabilities to provide such support. Therefore, in Chapter 2, we proposed a machine-readable file specification used to describe the metadata schema for the MIs of any MP. This standard metadata schema format enabled repository users, the modelers themselves, to define the metadata fields for any arbitrary model program. The schema could be consumed and utilized in a scalable way, because it provided a standardized way to encode user-defined metadata schemas and was machine-readable. This model metadata framework was deployed on HydroShare, a web-based collaborative system designed to advance sharing, accessing, and discovering hydrologic and environmental data and models (Tarboton et al., 2014b, 2014a). While the use of MI metadata schema allowed for extension of the specific metadata support to any model, and modelers could accommodates their particular requirements for their applications, only MI metadata schemas for SWAT and MODFLOW, previously introduced by Morsy et al. (2017), were provided and we did not delve into the principles needed to build a schema for any other models.

Researchers from different backgrounds have tried to provide principles that can be used when designing metadata for data and models. One of the well-known works is done by Duval et al. (2002). In their research, they provided metadata design *principles*, deemed to be common across all domains of metadata for data or models. Following these principles when designing a metadata schema for any application assures semantic and machine interoperability regardless of the chosen metadata standard. In this research, therefore, our goal is to demonstrate an approach for creating a schema for an arbitrary environmental model based on general principles provided by literature. In particular, this work builds from prior work that presents a flexible model metadata schema. It is important to note that, while in this research we focus on model metadata frameworks designed for one science discipline (i.e., environmental and hydrologic modeling), the gained insights can be still used by researchers from a wide range of research disciplines.

This chapter describes both the design of a flexible and scalable model metadata schema for a specific environmental model and demonstrates its use. We first discuss a set of design principles that need to be taken into account when designing a MI metadata schema. Then the design process is explored, and the experimental hydrological modeling use case is described. The Results of the study present the developed MI metadata schema for the case study model and its integration in to the HydroShare data and model repository system. We discuss the challenges for organizing the metadata schemas that are expected to be developed by the modeling community are discussed. Finally, we summarize the main contributions of this research and present possible research tasks to further improve model metadata needed to advance computational reproducibility and scientific transparency.

3.2 Methodology

Here we first investigate a set of principles that are used when designing a MI metadata schema and then the process of designing a metadata schema. Finally, an example use case of a hydrological modeling study is explained for which a MI metadata schema will be developed.

3.2.1 General principles for designing the MI metadata schema

Duval et al. (2002) lists a set of principles that are deemed to be common to all domains of metadata and which might inform the design of any metadata schema or application. Among the principles that they provided, three of them, modularity, extensibility, and refinement, are applicable to our MI metadata schema.

3.2.1.1 Modularity

Duval et al. (2002) argue that to make metadata modular, metadata elements from different schemas can be integrated in a syntactically and semantically interoperable manner to form a new metadata schema. This means a proposed MI metadata schema should be modular so that other researchers when creating their own MI metadata schema can learn from the existing practices by others, instead of starting from scratch. Also, if applicable, the notion of namespace, a formal collection of terms managed according to a policy or algorithm, can boost modularity of the metadata schema in a more formal way. Here we seek to identify the MI metadata elements that are common to a variety of hydrologic and environmental models and tries to generalize the findings so that they can be used for a wide range computational models.

3.2.1.2 Extensibility

As per the Duval et al. (2002) definition, metadata systems must be extensible in a way that they can take into account the specific needs of a given application. There are some metadata that are generally common between most metadata schemas (e.g., the concept of owner of an information resource) while other metadata are only specific to a particular science domain (e.g., watershed area in hydrology). This emphasizes the need for a base or core schema that captures the common metadata elements, and allows for extension by adding additional metadata based on the needs of a specific domain or an application within that domain.

In the case of environmental models, HydroShare has provided a base architecture that utilizes metadata to describe a resource (e.g., creator, or owner of the resource) and what it might contain, e.g., MI (e.g., spatial and temporal coverage metadata), or MP (e.g., software version metadata). The MI metadata schema introduced in Chapter 2 provides the opportunity to extend the metadata framework for any given hydrologic and environmental model. In this way, a MI metadata schema can benefit from using the common metadata among most hydrologic and environmental models while if needed have new elements specific to the model or the specific application based on the needs.

3.2.1.3 Refinement

The degree of detail needed for metadata can be different across different applications. As per the Duval et al. (2002) definition, a metadata schema should be designed in a way that it allows the schema designers to choose a level of detail according to their application and need. In this way, sufficient detail can be provided and unnecessary metadata will be avoided. One approach to help achieve refinement is using a controlled vocabulary which only allows for specifying a given metadata element from a particular set of values. This will enhance semantic interoperability across applications through relying on a common value set provided by the controlled vocabulary. Allowing sub-elements (metadata terms being divided into or element/sub-element aka parent/children sets) also helps the refinement of the metadata schema.

3.2.2 Process of designing metadata schema

In order to design a metadata schema for model instances, the steps shown in Figure 3.1 need to be taken. These steps are based on previous studies, Coyle and Baker (2008) and Ochiai et al. (2014), as well as our own experience.

The designer of the metadata schema first needs to define the metadata schema goal (Step 1). For example, in our case, the schema being developed is used to describe the input files or MIs of a specific hydrologic model. Then, given the application the schema is being developed for and using the existing practices by other researchers in the fields, the designer needs to specify the metadata elements and if there are needed sub-elements (Step 2). This requires the designer to have enough experience with the model for which the schema is begin developed for and detailed knowledge of the model structure, e.g., where input files are defined in the model. By using existing practices and standards, the metadata schema designer can use common elements and terminology across their science domain ("Where to start — advice on creating a metadata schema," 2014). In this way, the interoperability between different schemas within a domain increases.

Once the elements and sub-elements are decided, the designer needs to determine the attributes and constraints of them. For example, an element will be subject to min/max value constraints or can be only selected from a controlled vocabulary (Step 3). The next step is to convert the rough metadata schema developed through steps 1-3 to a metadata structure (e.g., JSON format) (Step 4). In this step, the designer needs to follow the syntax required by the schema format they are using. After converting the schema to metadata structure, the designer needs to validate the developed structure using online validation tools (e.g., https://www.jsonschemavalidator.net/) (Step 5). In this way, the designers of the metadata schema might discover issues in their metadata schema such as a syntax error or some unsatisfied requirements. Therefore, they will avoid any errors before integrating the validated structure into the data repository, HydroShare in our case (Step 6). Once the metadata schema is successfully integrated into the data repository through uploading, the original designer and other users can start using the schema to generate metadata elements for a MI of the given MP and populate them. Finally, if needed, the designer can start over the process (go to Step 1) and refine and improve the schema.

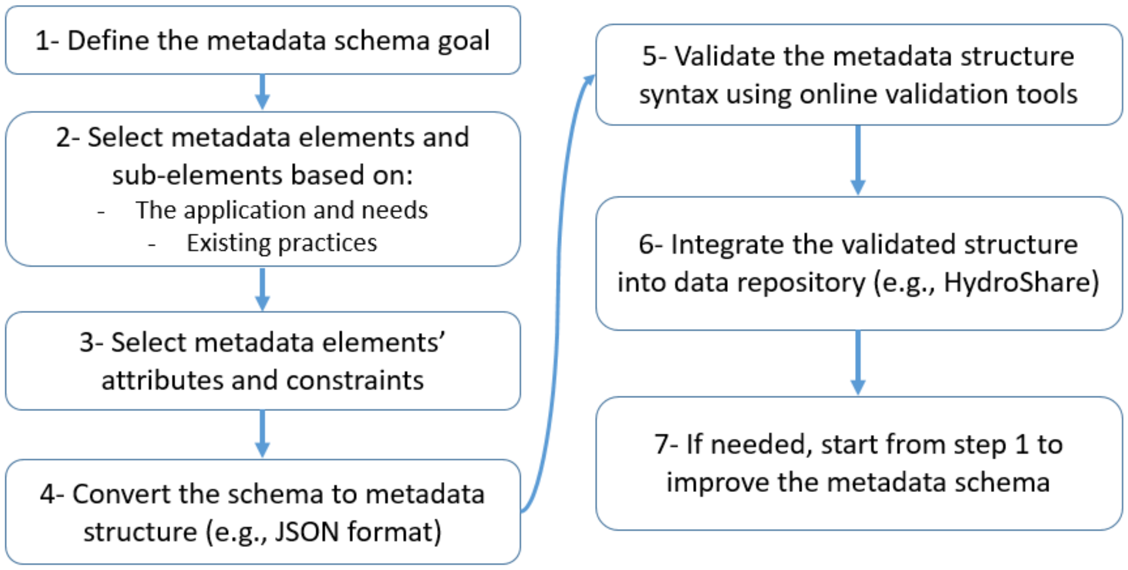

1- Define the metadata schema goal
2- Select metadata elements and sub-elements based on:
- The application and needs
- Existing practices
3- Select metadata elements' attributes and constraints
4- Convert the schema to metadata structure (e.g., JSON format)
5- Validate the metadata structure syntax using online validation tools
6- Integrate the validated structure into data repository (e.g., HydroShare)
7- If needed, start from step 1 to improve the metadata schema

Figure 3.1. Design process of a metadata schema

3.2.3 Model Metadata Design Case Study

In this section, we first discuss the hydrologic model for which we chose to build a MI metadata schema. Then, the web-based repository used to deploy the metadata schema is explained. Finally, we describe a use case that uses the selected hydrologic model and shares it on the web-based repository.

3.2.3.1 Selected hydrologic model: SUMMA

In this study we chose the Structure for Unifying Multiple Modeling Alternatives (SUMMA) (Clark et al., 2015b) as the case study model. SUMMA is a general environmental model that can be used to evaluate multiple model representations of hydrologic processes in a controlled and systematic way. SUMMA uses six configuration files to manipulate input files, model methods, and control the output files: (1) File Manager, (2) Decisions, (3) Forcing File List, (4) Model Output, (5) Param Trial, and (6) Local Attribute files. Model users can find the selected options for model input files, model methods, and output files in these configuration files.

3.2.3.2 Data repository: HydroShare

Tarboton et al. (2014b, 2014a) defined the design goal of HydroShare as to advance sharing, accessing, and discovering of hydrologic and environmental data and models. A "resource" in the context of HydroShare is the basic digital content unit that is being shared, and users can upload their data and/or model files and provide their corresponding metadata as a "resource". A "resource" can contain data as different "aggregation" types. As mentioned in Chapter 2, file aggregations are specific content types supported by HydroShare, extending Dublin Core metadata to provide new metadata elements that support functionality specific to common hydrologic datasets. Aggregations consist of several related files visualized with a single icon in HydroShare's file browser and may also have content-level metadata and specialized functionality available with HydroShare web apps.

Model Instance (MI) content aggregation represents the input (and optionally output) files while Model Program (MP) content aggregation includes the modeling software (Figure 3.2). Model Instance metadata schema is used to generate metadata fields for the MI based on the schema (green boxes in Figure 3.2).

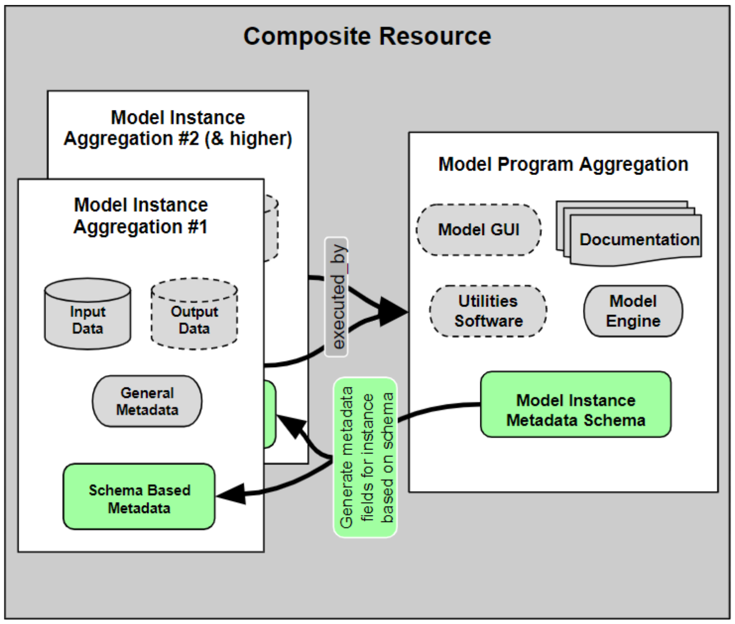

Composite Resource
Model Instance Aggregation #2 (& higher)
Model Instance Aggregation #1
Input Data
Output Data
General Metadata
Schema Based Metadata
executed_by
Generate metadata fields for instance based on schema
Model Program Aggregation
Model GUI
Documentation
Utilities Software
Model Engine
Model Instance Metadata Schema

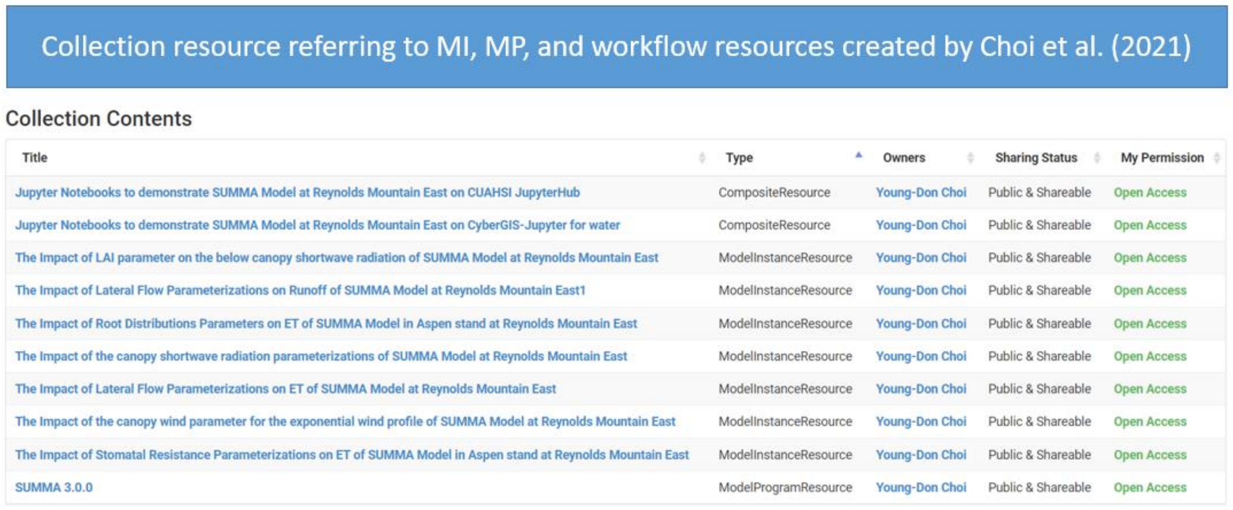
Figure 3.2. Key components of the MP and MI content aggregations in a HydroShare resource (source: adapted from Chapter 3 of this dissertatio)

3.2.3.3 Example model instance

To demonstrate our approach, we use the data and model from Choi et al. (2021d) who shared their modeling experiments as a HydroShare “collection resource” (Choi, 2020a). In their work, they used a set of hydrologic modeling experiments in Reynolds Mountain East Area in the Reynolds Creek Experimental Watershed in Idaho, USA described by Clark et al. (2015c). Choi et al. (2021d) used these modeling experiments with the purpose of reconstructing them and sharing them openly to make them easier to reproduce. However, when Choi et al. (2021d) shared their data and model, the new model metadata design allowing for sharing MI and MP content aggregations in a single HydroShare resource, and the MI metadata schema were not developed. Therefore, they used the old design of model resources to share the modeling experiments (Choi et al., 2021d) which led them to create a collection resource holding multiple resources with different resources types. Figure 3.3 shows the collection resources developed by Choi et al.

(2021d) that refers to: 1- two resources that each hold modeling workflows (i.e., Jupyter Notebooks), 2- seven MI resources, and 3- one MP resource.

The old metadata design by Morsy et al. (2017) and its implementation in HydroShare lacked a MI metadata schema specific to SUMMA; therefore, they were not able to describe their MIs with metadata specific to SUMMA. The framework introduced in Chapter 2 of this dissertation, which was implemented in HydroShare, provides a great motivation to use Choi et al. (2021d)'s HydroShare resources to show how the MI metadata schema capability developed in Chapter 2 as well as MI metadata schema developed in this study can be utilized. Therefore, in the results section, we will create a new HydroShare resource that presents Choi et al., (2021d)'s resources (Choi, 2020a) in an updated form to make use of shareability, accessibility and reusability facilitated through the MI metadata schema.


Collection resource referring to MI, MP, and workflow resources created by Choi et al. (2021)

Collection Contents

Title	Type	Owners	Sharing Status	My Permission
Jupyter Notebooks to demonstrate SUMMA Model at Reynolds Mountain East on CUAHSI JupyterHub	CompositeResource	Young-Don Choi	Public & Shareable	Open Access
Jupyter Notebooks to demonstrate SUMMA Model at Reynolds Mountain East on CyberGIS-Jupyter for water	CompositeResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of LAI parameter on the below canopy shortwave radiation of SUMMA Model at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of Lateral Flow Parameterizations on Runoff of SUMMA Model at Reynolds Mountain East1	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of Root Distributions Parameters on ET of SUMMA Model in Aspen stand at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of the canopy shortwave radiation parameterizations of SUMMA Model at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of Lateral Flow Parameterizations on ET of SUMMA Model at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of the canopy wind parameter for the exponential wind profile of SUMMA Model at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
The Impact of Stomatal Resistance Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East	ModelInstanceResource	Young-Don Choi	Public & Shareable	Open Access
SUMMA 3.0.0	ModelProgramResource	Young-Don Choi	Public & Shareable	Open Access

Figure 3.3. SUMMA use case: A HydroShare collection resource (Choi, 2020a) referring to seven MI, one MP, and two workflow resources by Choi et al. (2021d)'s hydrologic study (old design)

3.3 Results

3.3.1 Developed schema for SUMMA

The metadata schema to describe MIs linked to a SUMMA MP are shown in Figure 3.4. Four metadata elements, model objective, model input, model method, and simulation type, are selected to create a modular metadata schema. The reason to choose these four elements was to describe hydrologic and environmental models in a modular way. This was by exploring existing practices including the SWAT MI metadata schema, previously developed by Morsy et al. (2017). The terms were selected to be generic and applicable to other hydrologic models.

Model Objective metadata is used to describe the objectives of conducting a simulation, e.g., hydrology, climate change, and/or water quality. There are many common model objectives across different studies, therefore, this metadata element can benefit from a list of pre-defined objectives from which a user can select their objectives. Also, as there may be model objectives that are not provided in the list of the pre-defined objectives, a user can still provide their own objective. Model input seeks to describe the input files used in the simulation. Because there are different input files, this metadata element will have sub-elements each of which describe a single input file. Using sub-elements allows for refinement of the model metadata schema. Two examples of such sub-elements are watershed area and warm-up period metadata elements. Model methods describe the different methods used in the simulation and have sub-elements. Two examples of such sub-elements for the SUMMA model are the form of Richard’s equation to be used and stomatal resistance function. Simulation type specifies whether the simulation is to be used for sensitivity analysis, or calibration, or for a normal simulation of a calibrated model. Therefore, it can benefit from a controlled vocabulary that allows the user to choose from the list of the three simulation types. Also, if the simulation type is calibration, a new metadata element, i.e., calibration description, is generated. Calibration description allows the user to describe the calibration by providing two sub-elements, i.e., Algorithm (e.g., NSGA II) or the objective function(s) used to perform the calibration. The sub-elements of the model input and model method are considered as optional, so that the barrier to entry remains low.

Different attributes of an element of a MI metadata schema in a JSON format, including its definition, whether it is mandatory or optional, and any additional requirements are shown in Table 3.1. Among these attributes, there are four attributes that are mandatory to be provided by the designer of the metadata schema (represented by “M”), i.e., Name, Title, Type, and Property order, while the rest are optional (represented by “O”). The attribute “Name” has additional requirements required by the HydroShare implementation: it should have only alphanumeric characters and it must start with an alphabet character. The attribute “Title” is used as the label of the element in the user interface of the data repository. The attribute “Type” specifies the data type that is allowed for the metadata field. It can be of either string, number, object, or Boolean type and additional requirements may apply depending on the data type being used. While the attribute “Property order” is not a fundamental attribute of a metadata element, it is a number that determines the order by which an element is shown in the user interface.

If a metadata element needs to have sub-elements, an attribute called “Properties” can be used to include the sub-elements’ attributes. If a designer needs to make a list of controlled vocabulary from which a value may be selected for a metadata element, they need to use the attribute “enum” which also makes the data entry easier compared to typing it out manually. The metadata attributes “Default”, “minimum”, and “maximum” are also optional and may be respectively used to specify the default value, minimum (for data type “number”), and maximum (for data type “number”) acceptable value for a metadata element.

The developed JSON file for the SUMMA MI metadata schema is shown in Figure 3.5. In Figure 3.5a, the four metadata elements (i.e., model objective, simulation type, model input, and model output) as well as the calibration description (the conditional metadata field if the simulation type “calibration” is chosen) are shown. In Figure 3.5b, a snippet of the model input is provided which shows two of its sub-elements, warm-up period value and precipitation time step type. The Warm-up period’s value type is number, default and minimum value are both provided as 0.

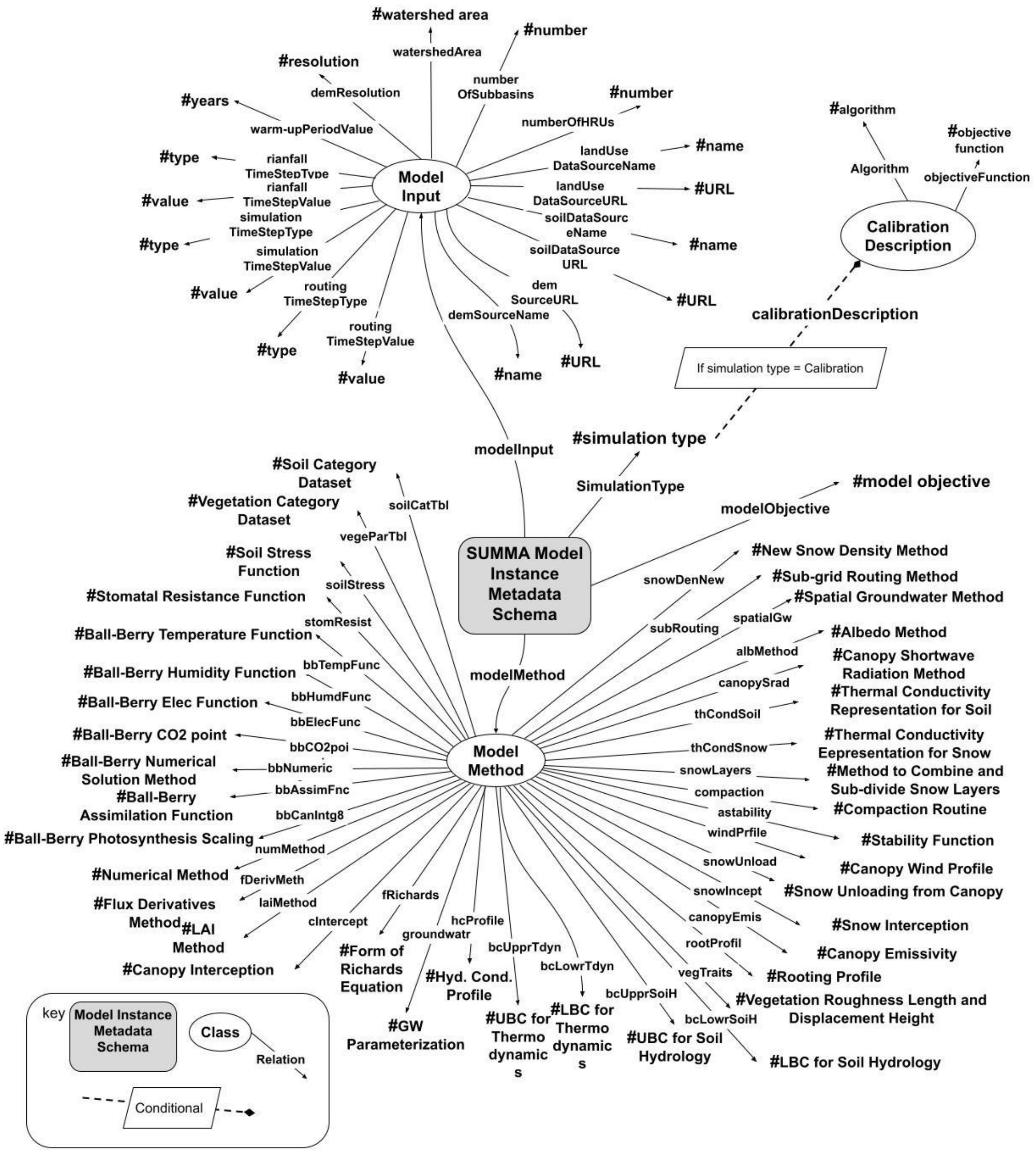
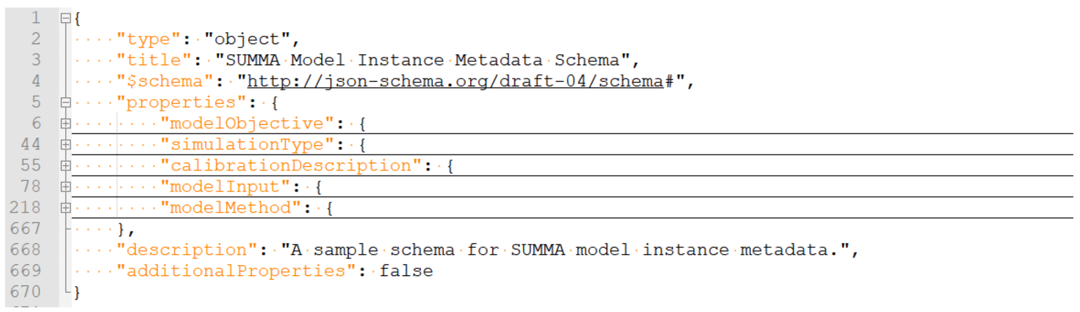


Figure 3.4. Developed MI metadata schema for SUMMA (metadata elements are not populated)

Table 3.1. Element's attributes and explanation using JSON schema (Implemented in HydroShare)

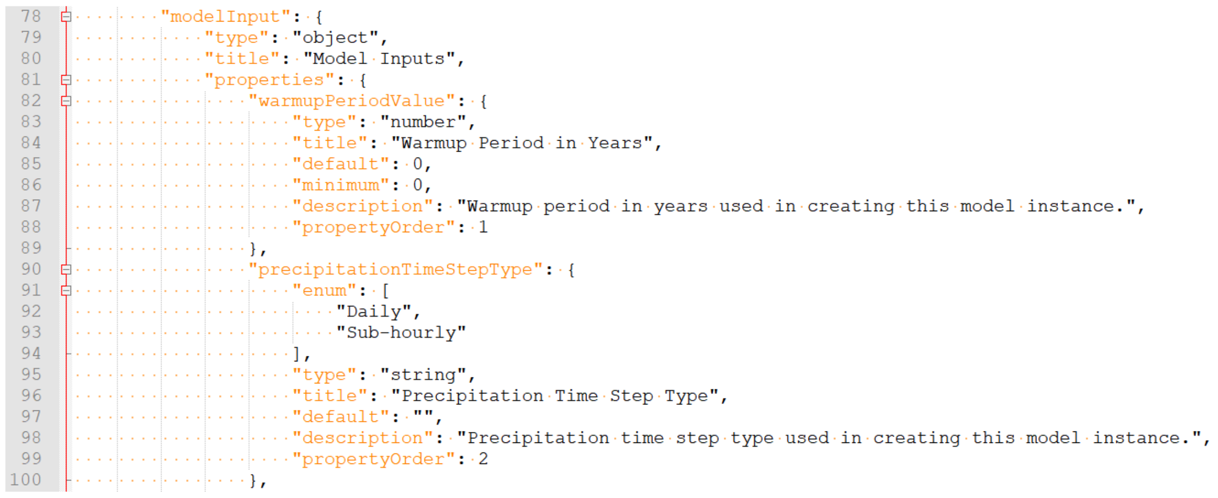
	Element's Attribute	Definition	Obligation to Use (M or O)	Additional Requirements
1	Name	The name of the element (used by machine to tag the element)	M	The following constraints are needed for XML/RDF encoding of model instance metadata based on metadata schema: 1. only alphanumeric characters are allowed 2. must start with an alphabet character
2	Title	The title of the element shown in the user interface (used as a label for that form field when the form is generated from the schema)	M	
3	Type	The type of data that the element is, either of: string, number, object, or Boolean	M	1. Any object type property must have the attribute 'additionalProperties' set to false. This is needed in order to prevent users from adding new form fields when editing metadata for model instance aggregation 2. To use a checkbox for a Boolean type field, attribute 'format' needs to be set to checkbox
4	Properties	Whether the element has sub-elements	O	Nested objects are NOT allowed: A field of type 'object' can't contain a sub-field of type 'object'. This ensures the schema based form does not get too complicated. A field/attribute of type 'object' can still contain other fields/attributes that are NOT of type 'object'
5	Description	The meaning of the data element (used to describe the purpose of the field which is used as help text and displayed below the form field when the form is generated based on the JSON schema)	O	
6	Property order	A number that determines the order by which an element is shown in the user interface	M	
7	Default	The default value of the element	O	
8	Minimum	The minimum acceptable value of the element (for type number)	O	
9	Maximum	The maximum acceptable value of the element (for type number)	O	
10	enum	Make a list of controlled vocabulary from which a value may be selected for an attribute	O	
11	format	Used for checkbox	O	
12	Additional properties	Used to control the handling of extra properties whose names are not listed in the properties keyword	O	
13	required	Users can make a field/attribute "required" by specifying that at the root level of the JSON schema.	O	

(a)



```
{
    "type": "object",
    "title": "SUMMA Model Instance Metadata Schema",
    "$schema": "http://json-schema.org/draft-04/schema#",
    "properties": {
        "modelObjective": {
        "simulationType": {
        "calibrationDescription": {
        "modelInput": {
        "modelMethod": {
    },
    "description": "A sample schema for SUMMA model instance metadata.",
    "additionalProperties": false
}
```

(b)



```
        "modelInput": {
            "type": "object",
            "title": "Model Inputs",
            "properties": {
                "warmupPeriodValue": {
                    "type": "number",
                    "title": "Warmup Period in Years",
                    "default": 0,
                    "minimum": 0,
                    "description": "Warmup period in years used in creating this model instance.",
                    "propertyOrder": 1
                },
                "precipitationTimeStepType": {
                    "enum": [
                        "Daily",
                        "Sub-hourly"
                    ],
                    "type": "string",
                    "title": "Precipitation Time Step Type",
                    "default": "",
                    "description": "Precipitation time step type used in creating this model instance.",
                    "propertyOrder": 2
                },
```

Figure 3.5. The developed JSON file for the SUMMA MI metadata schema

3.3.2 Integration into the HydroShare system

The contents of the HydroShare resource (Maghami, 2022) that we created are shown in Figure 3.6. It contains the MIs, MP, and model workflows in a single resource (Choi, 2022) originally shared as five MI resources, one MP resource, and two composite resources included in a HydroShare collection resource (Choi, 2020a). The model workflows that were shared as separate resources are now shared in the same resources in separate folders to demonstrate the uselessness of the new framework design and implementation.

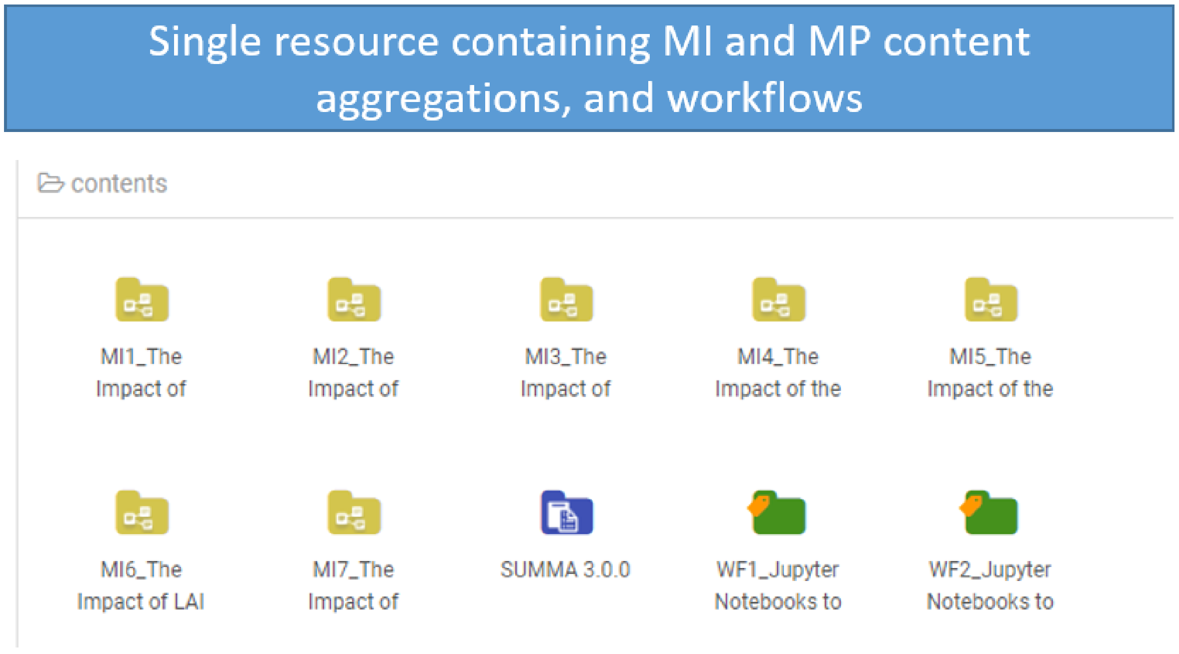
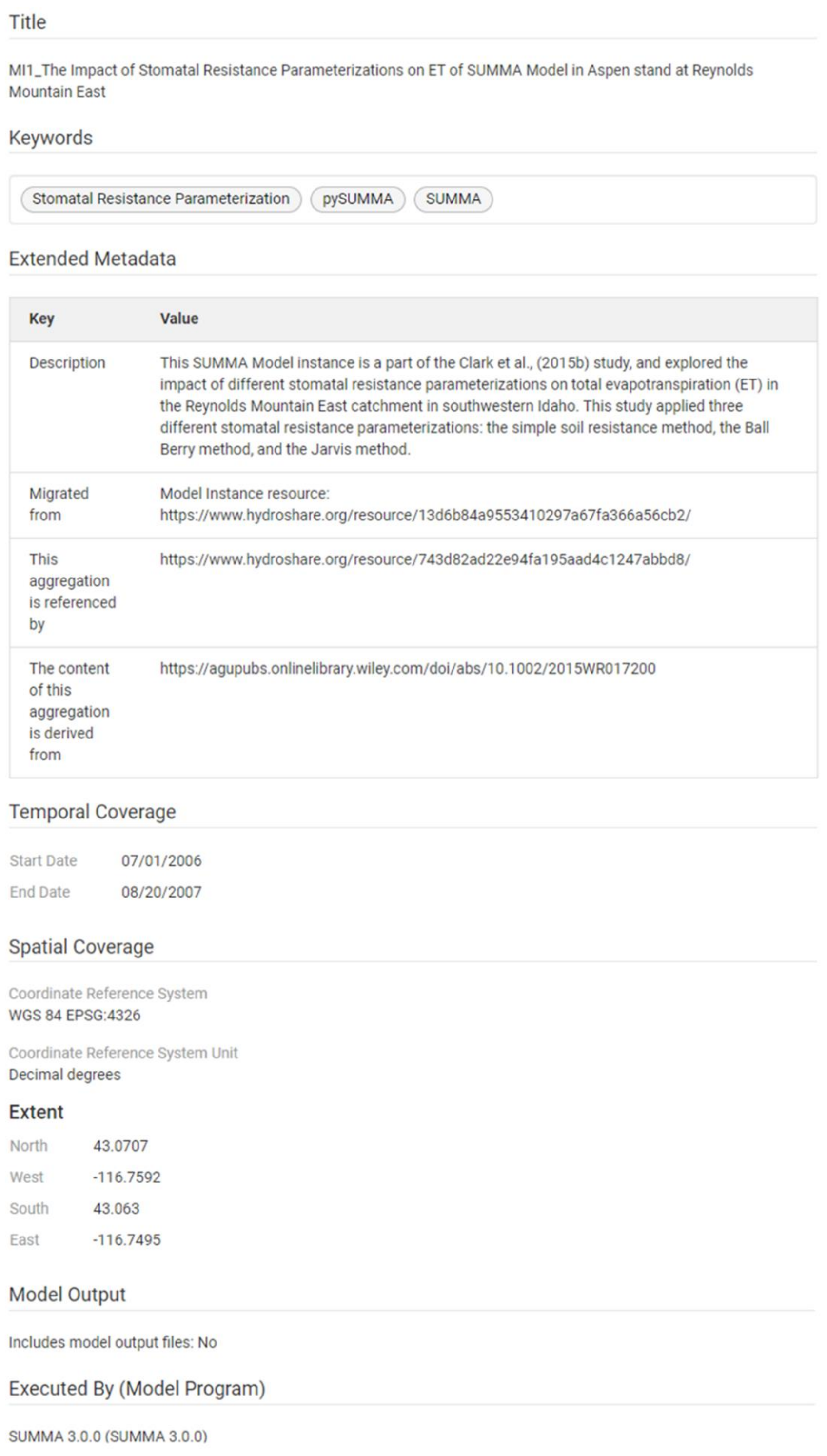


Figure 3.6. Results for SUMMA use case: single HydroShare resource holding seven MI and one MP aggregations and two workflows created by this study

Screenshots of a HydroShare resource that presents the metadata elements used to describe one of the MIs created in the HydroShare resource (Maghami, 2022) are shown in Figures 3.7 and 3.8. These metadata elements are the generic metadata elements used to describe a MI (Figure 3.7) or the schema-based metadata which are generated using the MI metadata schema developed through this study (Figure 3.8). Figure 3.7 shows the title of the MI aggregation, keywords, the linked MP, etc. It also shows the extended metadata used to include the metadata elements that were used in the original MI resource created by Choi (2020b), e.g., “Description” referring to the abstract of that resource, and “Migrated from” to show the original HydroShare resource. Figure 3.8 illustrates the schema-based metadata, including “Model Inputs” (e.g., Watershed Area = 32.7), “Model Method” (e.g., Albedo Method = varDecay), “Model Objective” (set as Hydrology), and “Simulation Type” (set as “Normal Simulation”).



Title

MI1_The Impact of Stomatal Resistance Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East

Keywords

Stomatal Resistance Parameterization | pySUMMA | SUMMA

Extended Metadata

Key	Value
Description	This SUMMA Model instance is a part of the Clark et al., (2015b) study, and explored the impact of different stomatal resistance parameterizations on total evapotranspiration (ET) in the Reynolds Mountain East catchment in southwestern Idaho. This study applied three different stomatal resistance parameterizations: the simple soil resistance method, the Ball Berry method, and the Jarvis method.
Migrated from	Model Instance resource: https://www.hydroshare.org/resource/13d6b84a9553410297a67fa366a56cb2/
This aggregation is referenced by	https://www.hydroshare.org/resource/743d82ad22e94fa195aad4c1247abbd8/
The content of this aggregation is derived from	https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015WR017200

Temporal Coverage

Start Date	07/01/2006
End Date	08/20/2007

Spatial Coverage

Coordinate Reference System
WGS 84 EPSG:4326

Coordinate Reference System Unit
Decimal degrees

Extent

North	43.0707
West	-116.7592
South	43.063
East	-116.7495

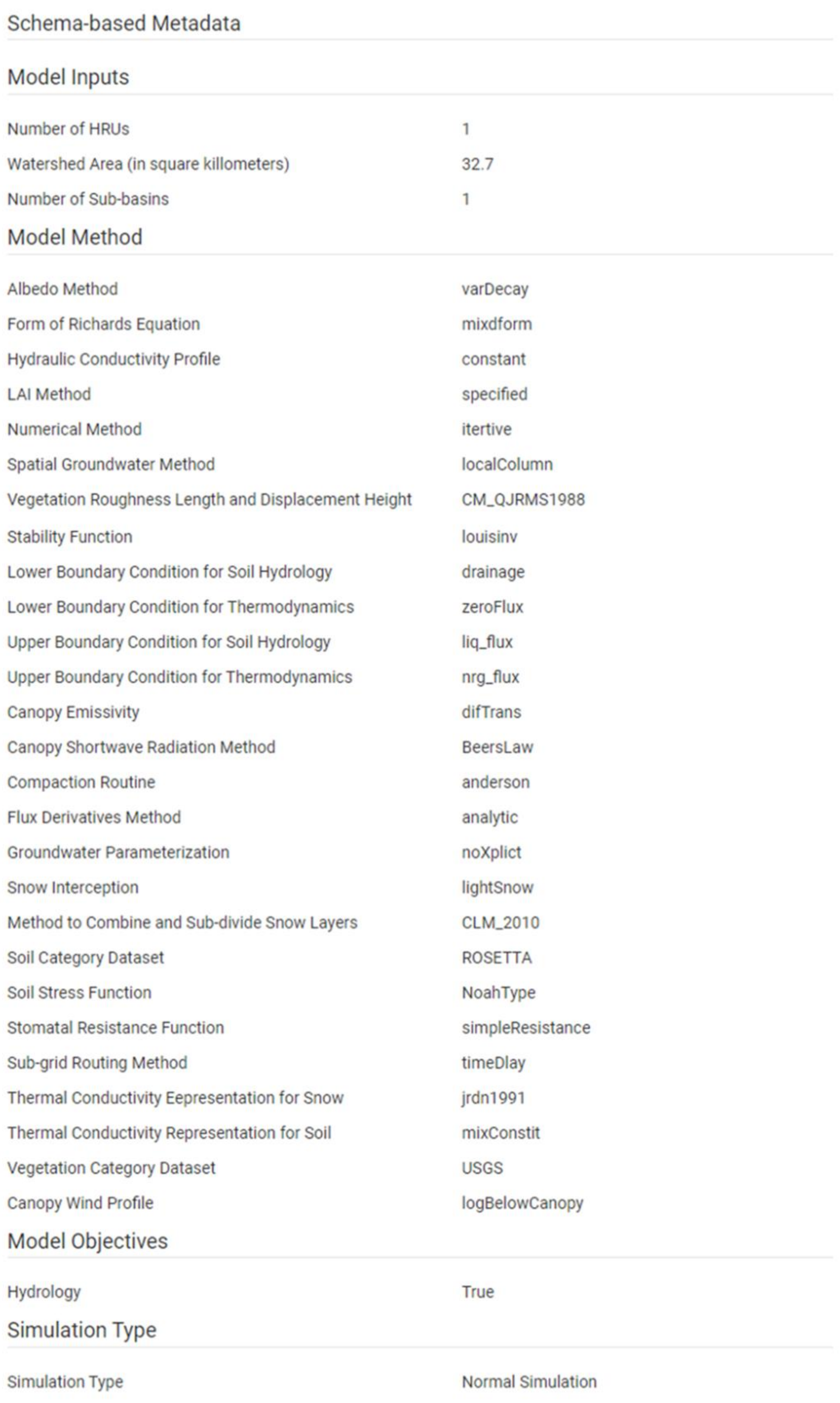
Model Output

Includes model output files: No

Executed By (Model Program)

SUMMA 3.0.0 (SUMMA 3.0.0)

Figure 3.7. Model Instance content aggregation generic metadata for MI1_The Impact of Stomatal Resistance Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East



Schema-based Metadata

Model Inputs

Number of HRUs	1
Watershed Area (in square killometers)	32.7
Number of Sub-basins	1

Model Method

Albedo Method	varDecay
Form of Richards Equation	mixdform
Hydraulic Conductivity Profile	constant
LAI Method	specified
Numerical Method	itertive
Spatial Groundwater Method	localColumn
Vegetation Roughness Length and Displacement Height	CM_QJRMS1988
Stability Function	louisinv
Lower Boundary Condition for Soil Hydrology	drainage
Lower Boundary Condition for Thermodynamics	zeroFlux
Upper Boundary Condition for Soil Hydrology	liq_flux
Upper Boundary Condition for Thermodynamics	nrg_flux
Canopy Emissivity	difTrans
Canopy Shortwave Radiation Method	BeersLaw
Compaction Routine	anderson
Flux Derivatives Method	analytic
Groundwater Parameterization	noXplict
Snow Interception	lightSnow
Method to Combine and Sub-divide Snow Layers	CLM_2010
Soil Category Dataset	ROSETTA
Soil Stress Function	NoahType
Stomatal Resistance Function	simpleResistance
Sub-grid Routing Method	timeDlay
Thermal Conductivity Eepresentation for Snow	jrdn1991
Thermal Conductivity Representation for Soil	mixConstit
Vegetation Category Dataset	USGS
Canopy Wind Profile	logBelowCanopy

Model Objectives

Hydrology	True

Simulation Type

Simulation Type	Normal Simulation

Figure 3.8. Model Instance content aggregation schema-based metadata for MI1_The Impact of Stomatal Resistance Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East

3.4 Discussion

3.4.1 Categorizing metadata based on their harvestability

Some metadata are easily harvestable using the input files. For this metadata, an automated code can easily extract the metadata values from the input files. One example of such metadata is the DEM resolution. However, other metadata are not recoverable using the input files and the user needs to specify them. Examples of this metadata are: What was the purpose of this study? Or, what was the simulation type? Normal simulation, sensitivity analysis, or calibration. The second type is the metadata that serves as barrier to put the model in (because they cannot be auto extracted from the input files) but makes the crucial part of a schema providing a high-level description of the MI while the metadata from the first type help to provide more detailed information on the MI.

3.4.2 Model setup code: a metadata element to consider

This paper focuses on describing the model software itself. Future research could explore metadata for describing the software used to prepare model instances, i.e. the software used to generate the SUMMA (or any other hydrologic and environmental model) required model input files (e.g., https://github.com/CH-Earth/CWARHM). This metadata could be an extension of the MI metadata schema or it could be a separate metadata schema. It would be helpful to include the model setup source code or a link to an external repository as part of the MI metadata schema because that allows others to not only find out how the model was used, but also how its input data were generated. This has two main benefits: (1) it tracks the full process and provides a record of any choices made during the model setup, and (2) in cases where the MI covers a large domain and sharing all input data may become unwieldy, it shares the code that was used to generate the input data so that there is a traceable record of what went into the model.

3.4.3 Strategies for managing the MI metadata schemas

By the time the modeling community starts creating their own MI metadata schemas, there will be a need to manage the MI metadata schemas that might be developed. In this way, designers will be able to find, reuse, and adapt the existing practices. For this reason, there should be a mechanism to enable discoverability and promote reuse of the created MI metadata schemas. This mechanism needs to consider providing some metadata about the schema itself. These metadata, at the very least, include creators of the schema, the schema version, why the schema was developed, and the model name (Table 3.2). This ties back to the first common metadata principle mentioned by Duval et al. (2002), modularity, as it lets other researchers reuse and adapt the developed MI metadata schemas. Creating a database to manage the MI metadata schema was beyond the scope of this work, but it can be implemented in future research.

Table 3.2. Metadata elements to be considered when describing a MI metadata schema, proposed to manage the schemas that will be created by the modeling community

	Element	**Definition**	**Obligation to Use (M or O)**
1	Creator of Schema	Who created the schema	M
2	Schema version	The version of the schema	M
3	The goal of schema	For what goal this schema was developed	O
4	Model	What environmental or hydrologic model this schema is developed for	M

3.5 Conclusions

In this research, we designed a MI metadata schema for a SUMMA model and explained the steps needed for that. This practice can be followed to design MI metadata schema for other hydrologic and environmental models. Using the MI metadata schema, the user can define the semantics (i.e., properties such as title and description of the field), and rules for the extended metadata fields content (i.e., properties such as data type and default value, whether the field is mandatory, optional or conditional (implications), whether the field has nested fields inside it, and whether the field must be chosen from a set of controlled values, whether it is multiple checkbox fields). We tried to only accommodate a sufficient level of detail in the proposed framework. The focus was on including high-level metadata used to describe a MI. In this way, we ensured that the MI is described in a generic way while the barrier to entry still remains low as the very detailed metadata are kept out of the MI metadata schema. Users are able to include metadata beyond the metadata proposed by this framework for cases in which a particular set of metadata is important to them based on their needs and applications.

To select the metadata elements, we used the common metadata principles highlighted by literature when designing any metadata system and showed how these principles are used. We also provided the sequential steps needed to design the MI metadata schema so that the key points are highlighted and the designers of any metadata schemas can understand how to get started. We integrated the developed schema into HydroShare and presented a use case where the MIs of a hydrologic model are described using the developed MI metadata schema. The MI metadata schema is totally generic and independent of HydroShare, so the metadata designers should be able to use it in their own software.

Upon creation of a number of different metadata schemas for different models by modelers, there will be a need to organize the metadata schemas so that they can be easily shared, found and reused. Therefore, a method for organizing the metadata schemas can be the topic of future research.

This chapter is in preparation for submission to a peer-reviewed journal. Current coauthors of the study Iman Maghami, Ashley Van Beusekom, Lauren Hay, Zhiyu Li, Andrew Bennett, YoungDon Choi, Jonathan L. Goodall, Bart Nijssen, David Tarboton

Chapter 4

Cyberinfrastructure for Reproducibility of Complex Hydrologic Modeling Studies

4.1 Introduction

Reproducibility, the ability to duplicate and verify previous findings, is a foundational principle in scientific research (Hutton et al., 2016). In hydrology, Melsen et al. (2017) argue that “conclusion-reproducibility” (replicating a study’s conclusions) may be more important than “bit-reproducibility” (exactly replicating an entire study) because hydrological theories need to be tested for different situations (beyond “bit-reproducibility”) by investigating conditions under which theories can be confirmed or falsified. Even so, “conclusion-reproducibility” goes beyond the simple sharing of code and data as open-source and online resources. The code and data must be accompanied by well-documented workflows with readable and reusable code (Chen et al., 2020). A further step is to provide open-source computational environments in which these workflows can be executed. Ensuring this reproducibility is a non-trivial task; it requires not only new cyberinfrastructure that includes dynamic resource provisioning, computational notebooks (e.g., Jupyter), and container technology (Hutton et al., 2016; Merkel, 2014), but also careful software engineering practices including software version-control and documentation.

A general strategy for creating modern cyberinfrastructure to support open and reproducible hydrologic modeling is described by Choi et al. (2021d). This approach integrates three main components: (1) online data repositories; (2) computational environments leveraging containerization and self-documented computational notebooks; and (3) Application Programming Interfaces (APIs) that provide programmatic control of complex computational models. As an example of this approach, Choi et al. (2021d) presented an implementation that used (1) HydroShare (www.hydroshare.org) as the online repository, (2) two different Jupyter instances, one hosted by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) and a second hosted by CyberGIS-Jupyter for Water, as the computational environments, and (3) pySUMMA, a Python wrapper for manipulating, running, managing, and analyzing SUMMA (Structure for Unifying Multiple Modeling Alternatives), as the model API.

Choi et al. (2021d) focused mainly on the system design and demonstrated their approach with a fairly simple modeling example. Reproducibility in computational hydrology, however, can present some difficult challenges when dealing with large-scale hydrologic studies (Hutton et al., 2016). These challenges mostly pertain to the use of “big data” and computationally expensive and time-consuming resources needed for reproducibility of complex hydrologic modeling studies. Hutton et al. (2016) notes that in these cases, new techniques are needed to ensure scientific rigor. In this paper, we provide an example of the overall system design outlined by Choi et al. (2021d) as applied to a large-scale hydrologic study by Van Beusekom et al. (2022) (hereafter referred to as the VB study). We develop the necessary cyberinfrastructure to reproduce this study for selected sub-domains and discuss the challenges and opportunities in ensuring that large-scale hydrologic studies are reproducible.

The VB study evaluated the effect of the temporal resolution of surface meteorological inputs (or forcings) on modeled hydrological fluxes and states for 671 basins across the contiguous United States (CONUS). It quantified the difference in hydrologic outcomes based on daily or sub-daily forcings for multiple model configurations and parameter values. Reproducing the entire VB study based solely on the input data and model code is challenging because it requires the installation and configuration of the modeling framework SUMMA (Clark et al., 2015b, 2015a), the data volumes are very large, and the model runs require High Performance Computing (HPC) resources. The complete VB study consisted of 704 6-year model runs for each of the 671 basins (or 2.8 million model years). SUMMA was implemented with a single hydrologic response unit for each basin (lumped model), resulting in a single output time series for each basin for each model configuration. For every model run, the output consisted of 14 hydrological variables, which required 6 MB per model simulation, or 2.834 TB for the entire study. While few researchers may be interested in reproducing the entire VB study, the more common use case may be a desire to repeat or extend the VB study for a subset of the basins. We want to enable others to reproduce the VB study for subsets of the original domain as a basis for doing additional research. For such an approach to be effective, it is not sufficient to provide the open-source SUMMA code and model input data; one must also provide the additional components described by Choi et al. (2021d), i.e., computational environments, models exposed through APIs, and documented model workflows to create a cyberinfrastructure that lowers the barrier to reproducibility.

To achieve this, we designed and implemented cyberinfrastructure to enable intuitive access to HPC computational environments and to support data transfers into and out of the HPC environment. Additionally, we provide a workflow that allows users to replicate parts of the study within their own computing environments. We also perform a workflow run-time performance analysis that compares different model scenarios by varying the size of simulations across different computing environments. This analysis may serve the users as a guide for the selection of the computing environment depending on the size of their simulations. The cyberinfrastructure provides a starting point for users to modify the hydrologic model setups, thus going beyond reproducibility into replication where one modeling methodology can be used with new input data (Essawy et al., 2020). The cyberinfrastructure may also serve as an educational resource by providing an intuitive way for students to perform complex hydrologic modeling studies. The data and cyberinfrastructure are provided through HydroShare to run on any basin with a SUMMA setup to assist the modeler in analyzing basins individually, making the methodology accessible, interoperable, reusable, and reproducible.

Previous research has started exploring the use of cyberinfrastructure for hydrologic modeling. Yang et al. (2010) underlines the importance of using HPC in computationally intensive geospatial sciences and hydrologic modeling. Essawy et al. (2016) developed server-side workflows for large-scale hydrologic data processing but they did not make use of HPC in their application. Lyu et al. (2019) used containerization and combined computational environments including HPC and high throughput computing (HTC) cyberinfrastructure to directly run the models using Jupyter notebooks. Gan et al. (2020a) integrated a hydrologic data and modeling web service with HydroShare as a data sharing system to show how this integration leads to a findable and reproducible modeling framework. Gichamo et al. (2020) used web-based data services to prepare input data for hydrologic models. Kurtz et al. (2017) introduced a cloud-based real-time data assimilation and modeling framework and showed how parallel processing can be used for complex hydrologic models in the cloud. However, none of Lyu et al. (2019), Gan et al. (2020a), Gichamo et al. (2020) and Kurtz et al. (2017) applied their methods on a computationally extensive hydrologic use case unlike the VB study. Therefore, the challenges and opportunities for complex hydrologic modeling remain unexplored.

The objectives of this paper are to (1) explore how the strategy and architecture presented by Choi et al. (2021d) can be extended to reproduce a data and computationally intensive hydrologic study, (2) provide the users with a guide on the selection of the computing environment depending on the size of their simulations, and (3) to identify and document implementation challenges and roadblocks.

The remainder of this chapter is organized as follows. In Section 4.2, we provide a brief overview of the VB study, the cyberinfrastructure, the model workflows, and the model scenarios used for a science use case subsetted from the VB Study as well as the model workflows run-time performance analysis. Section 4.3 provides results and discussion. The results focus on the modeling case study, and an analysis of the workflow run-time performance for different computing environments. Later in that section we discuss the opportunities and challenges learned from our development of the cyberinfrastructure to support our modeling workflows. Finally, our conclusions and recommendations are provided in Section 4.4.

4.2 Methods

The following sections present (1) an overview of the computational setup of the VB study (Section 4.2.1); (2) the cyberinfrastructure design and implementation to support reproducibility of the VB study (Section 4.2.2); (3) the model workflows that allow the user to reproduce all or parts of the VB study in an HPC environment or using CyberGIS-Jupyter for Water (CJW) Virtual Machine (VM) computing resources (Section 4.2.3); and (4) the model scenarios and run-time performance analysis (section 4.2.4).

4.2.1 Overview of the VB study

The VB study used 671 basins to study the effects of the temporal resolution of the meteorological forcings on hydrologic model simulations across the CONUS. The basins are part of the CAMELS dataset (Catchment Attributes and MEteorology for Large-sample Studies; (Newman et al., 2015b)), which is a large-sample hydrometeorological dataset across the CONUS consisting of input forcings, catchment attributes, and relevant historical streamflow records. The VB study used the Structure for Unifying Multiple Modeling Alternatives (SUMMA) (Clark et al., 2015b) to configure multiple model instances for each basin, representing eight different model configurations and 11 different sets of model parameter values. In addition, eight forcing datasets were constructed. In each of these forcing datasets one of the meteorological inputs was modified so that the diurnal cycle was replaced by the mean value over that day. Van Beusekom et al. (2022) performed 704 (8×11×8= 704) 6-year model runs for each CAMELS basin, consisting of one year of model initialization and five years of actual simulation. Model outputs for 14 simulated variables were stored to evaluate the sensitivity of the simulations to changes in model forcings, model configurations, and model parameters (Figure A1 and Table A1 in the Appendix). The VB study results demonstrated that (1) the effect of each forcing input on each model output varies by model output and model location, (2) the use of a particular parameter set may not be critical in determining the most and least influential forcing variables, and (3) the choice of model physics (i.e., using different model configurations) could change the relative effect of each forcing input on model outputs.

The VB study was run with scripts on the Cheyenne supercomputer (a 5.34-petaflops, high-performance computer built for the National Center for Atmospheric Research; https://www2.cisl.ucar.edu/resources/computational-systems/cheyenne), and it took a few days to complete the runs. For each basin, the output size for a single 6-year run was 6 MB. Thus, reproducing the entire study is computationally expensive and also requires large amounts of storage (704 runs × 671 basins × 6 MB = 2.834 TB). However, the structure of the model allows individual basins to be run independently. Here, we focus on a use case in which a researcher wishes to reproduce a subset of the VB study by analyzing one or a few basins within a cloud cyberinfrastructure environment.

4.2.2 Cyberinfrastructure design and implementation

Following the approach described in Choi et al. (2021d), we designed and implemented cyberinfrastructure (Figure 4.1) to replicate the VB study by integrating the (1) HydroShare online data repositories, (2) high-performance and CyberGIS-Jupyter for Water Virtual Machine (CJW VM) computational environments, and (3) a model API that can be utilized in scripts using Jupyter notebooks (here the pySUMMA API). The CJW VM computational environment relies on CJW cloud computing resources hosted on Jetstream cloud (Hancock et al., 2021; Stewart et al., 2015; Towns et al., 2014) and does not make use of High Performance Computing (HPC). With some additional work, the 'CJW VM computational environment' can also be hosted on other (non CJW) cloud services. Each of these three components is further explained in the following subsections.

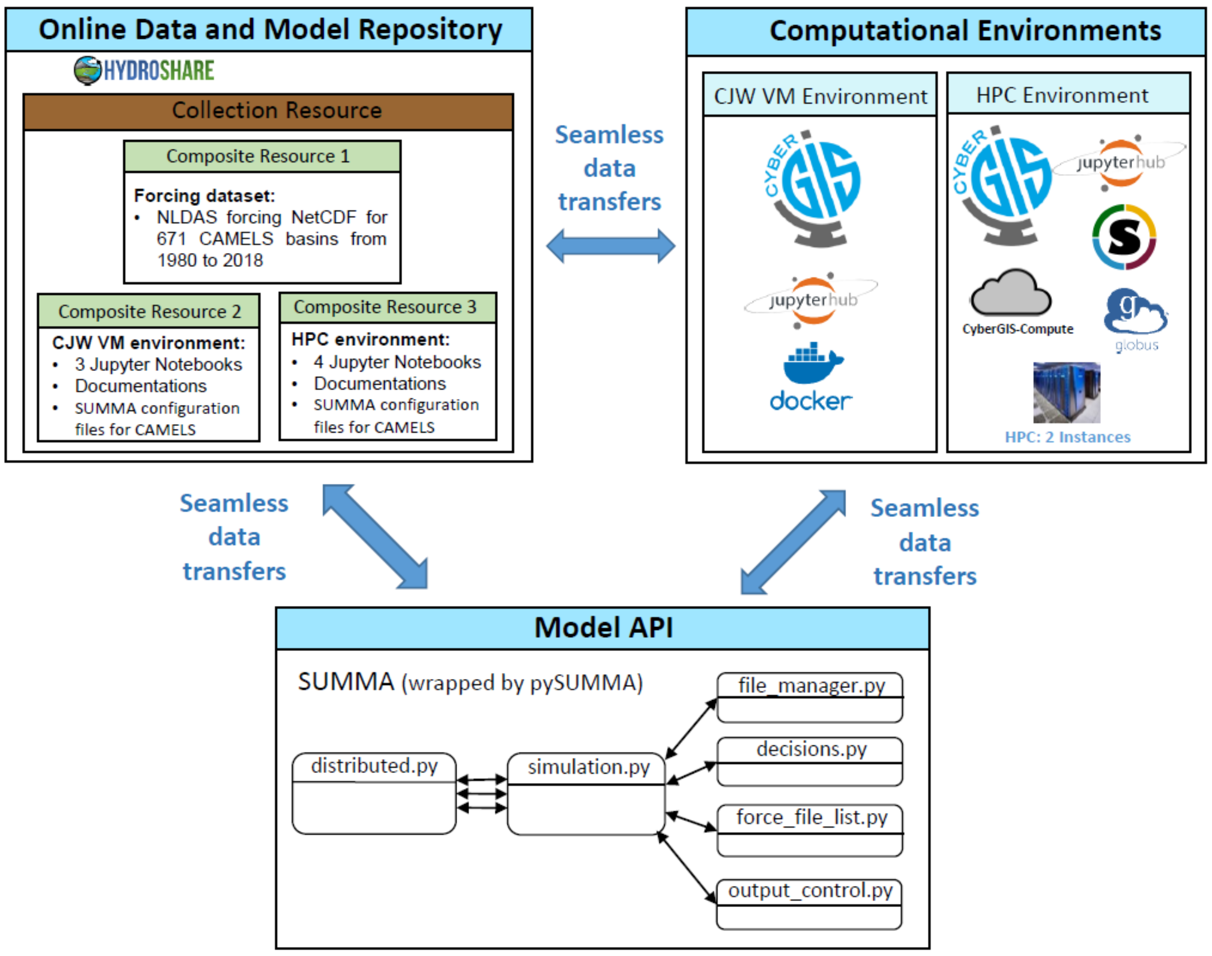


Figure 4.1. The general modeling approach presented by Choi et al. (2021) used by this study consisting of three primary components with seamless data transfers for open and reproducible environmental modeling

4.2.2.1 Online data repositories

We used HydroShare to store and share the data and modeling workflows for this study. HydroShare (www.hydroshare.org) is an online collaboration environment for sharing data, models, and code (Essawy et al., 2020; Horsburgh et al., 2016; Tarboton et al., 2014a). A collection resource in HydroShare, which can be found at Choi et al. (2021a), contains three resources holding the data, computational environment, and models (Figure 4.1).

The HydroShare resource holding the data (Mizukami and Wood, 2021) contains the forcing data set for the 671 CAMELS basins. The forcings are based on the hourly NLDAS-2 (North American Land Data Assimilation System; https://ldas.gsfc.nasa.gov/nldas/v2/forcing; NLDAS-2 is hereafter referred to as NLDAS). The original NLDAS hourly forcing data were created by NOAA on a 0.125 x 0.125 degree grid. To create hourly SUMMA model forcings, NLDAS outputs were spatially averaged over each of the 671 CAMELS basins and merged into one NetCDF file. With this format, an OPeNDAP server (https://earthdata.nasa.gov/collaborate/open-data-services-and-software/api/opendap/opendap-user-guide) can extract data for selected basins on the server, so that the user does not have to download the entire CONUS dataset to a local computer. HydroShare offers this capability via its THREDDS Data Server (TDS).

4.2.2.2 Computational environments

The developed computational environments were intended to provide a consistent software environment that was independent of each user's own operating system and software libraries and that made it possible to study a computationally expensive research problem. One computational environment was developed on an HPC resource to reproduce a problem more representative of challenges posed by the use of big-data in the VB study. A second environment was implemented on the CJW VM cloud service for studies for a more limited computational demand, e.g. a study of only a few basins. Reproducibility was facilitated by using containerization of the SUMMA model and the pySUMMA API with Docker (Merkel, 2014) for the CJW VM environment or Singularity in the case of the HPC environment (Kurtzer et al., 2017) along with a computational gateway interface to Jupyter IPython notebooks (pySUMMA and the notebooks are described in a later section). We avoided the use of Docker in the HPC environment because of security concerns, since Docker images can provide a means to gain root access to the system on which they are running (Kurtzer et al., 2017). The containerization and interface are hosted on the CJW scientific cloud service. An HPC option was added that allows the user to reproduce a problem more representative of the challenges posed by the use of big-data in the VB study. More importantly, it allows the user to study a particular basin to a level of detail that can further their own research. The Conda software package is used to manage the project specific computational environment on CJW, allowing the user to build a Python environment with the SUMMA model, pySUMMA API, and other computational dependencies. This is done by providing a kernel version for the project (CyberGIS Center HydroShare Development Team, 2022). Using this stable kernel, which captures all the required dependencies with their specific versions, ensures careful software version-control.

4.2.2.3 Application Programming Interface (API)

The model API pySUMMA was chosen to be part of the interactive tool. The pySUMMA API (Choi et al., 2021d) wraps the hydrologic modeling framework SUMMA (Clark et al., 2015b) and allows the user to script the use of the SUMMA model using Python. It facilitates model configuration and allows for local execution of the model by either using a Docker container or a locally compiled SUMMA executable (Choi et al., 2021d). With pySUMMA, a user can modify SUMMA input files and run SUMMA inside a Python script, as well as automatically parallelize runs and visualize output. In the simplest case the pySUMMA Simulation object wraps a single instance of a SUMMA simulation. For users who choose to analyze multiple basins at a time in the CJW VM environment, the notebook automatically will configure a pySUMMA Distributed object, which provides an interface to spatially distributed simulations and handles parallelism and job management under the hood. In this study, multiple SUMMA simulations are run in each basin, so a pySUMMA Ensemble object is used to manage multiple runs with different configurations. In the HPC computational environment a custom backend was written to handle parallelism using MPI, reducing the need for users to customize the configuration based on the type of job that they are running. A high-level description of pySUMMA is presented in Figure 4.1. The simulation.py enables the execution of the SUMMA model and along with file_manager.py, decisions.py, force-file_list, and output_control.py allows for manipulating SUMMA configuration files. The distributed.py enables the parallel execution of SUMMA.

4.2.2.4 Data management and transfer

The input data for this study consists of the SUMMA configuration files and the forcing data for the 671 CAMELS basins. The configuration files (e.g., geometries information for the 671 CAMELS basins along with their attributes such as hru_id) are shared within each of the two HydroShare resources holding the Jupyter notebooks (Choi et al., 2021b, 2021c). The forcing data are provided in a HydroShare resource (Mizukami and Wood, 2021).

The output files resulting from running the notebooks using the CJW VM and HPC computational environments are, (1) the NetCDF output files generated by the SUMMA simulations, (2) a NetCDF file recording the model performance for each basin as measured by the Kling-Gupta Efficiency (KGE) (Gupta et al., 2009), and (3) additional files created by the notebooks such as the figures that visualize the model results.

In the case of the CJW VM environment, after running the notebooks, all files are saved in the CJW platform and are directly accessible to the user. In the case of the HPC environment, the KGE results and other files created by the notebooks (e.g., figures) are automatically transferred to the CJW platform, but the NetCDF output files remain within the HPC environment to avoid transferring large volumes of model output (the size of the model output for the entire VB study was 2.834 TB).

To enable the user to transfer selected SUMMA NetCDF output files to the CJW environment for further analysis and long-term storage, the CyberGIS-Compute Service (Li et al., 2022) supports reliable high-performance large file transfers needed for HPC workflows through the Globus service (Chard et al., 2016; Foster, 2011). Figure 4.2 shows that data is transferred from HPC to the CJW using Globus without going through the job submission server. Globus is a software as a service, enabled by the cloud which provides its users with a secure reliable service for research data management. It enables the user to transfer datasets of any size between different storage options (personal computers, HPC, etc.) without users being required to be constantly logged in and watching (Chard et al., 2016).

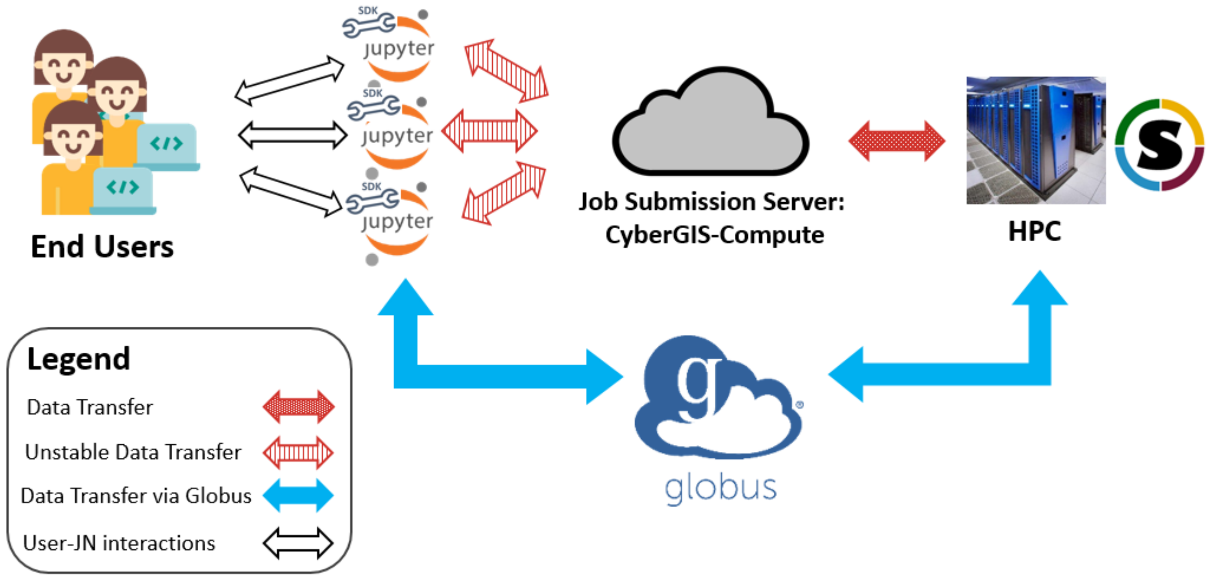


Figure 4.2. Large data transfer from HPC to CyberGIS-Jupyter for Water Jupyter on HydroShare through 1- the CyberGIS-Compute, 2- Globus

4.2.3 Model Workflows

As mentioned earlier, the model workflows allow the user to reproduce all or subsets of the VB study in an HPC environment or using CJW VM computational resources. The model workflows are documented in three (CJW VM) or four (HPC) Jupyter notebooks Table 4.1 shows the summary of the steps taken in each notebook while Figure A2 - A5 in the Appendix show more detailed information for notebooks 1-4. The first three notebooks for both the CJW VM and HPC environments focus on (1) preparing the input forcing for the model, and setting study basins and simulation period, (2) running the SUMMA model, and (3) exploring outputs to analyze the effect of each forcing variable in each basin. The HPC computational resource uses a fourth notebook to transfer large unprocessed output data from the HPC to CJW using GLOBUS. Notebooks 1 and 3 are very similar between the two HydroShare resources, and both CJW VM and HPC HydroShare resources use CJW VM computational resources to run these two notebooks. The second notebook differs for the two environments, and the difference is explained in section 4.2.3.2. These notebooks assist a modeler in analyzing CAMELS basins individually, providing information on forcings and output variables that are the most/least sensitive in their basin.

Table 4.1. Overview of the notebooks 1-4

#	Notebook Name	Goal	CJW VM or HPC
1	Preprocessing	Prepares forcings, and sets study basins and simulation period	Very similar between HPC and CJW VM environment
2	SUMMA execution	Runs the SUMMA model	Different versions for HPC and CJW VM environment
3	Post-processing	Explores outputs to find out effect of each forcing variable in each basin	Very similar between HPC and CJW VM environment
4	Use Globus to transfer big data	Transfer raw output from HPC to CJW using Globus service	Only for HPC environment

To use the HPC computational resource, the user must first obtain access to the HPC server. To do so, the user issues a request through HydroShare to use CJW. Once this access is granted, users are automatically given free access to two alternative HPC resources. The first HPC resource is the Virtual ROGER (Resourcing Open Geospatial Education and Research) HPC administered by the School of Earth, Society, and Environment at UIUC which is integrated with the Keeling compute cluster at UIUC (https://cybergis.illinois.edu/infrastructure/hpc-user-guide/). The second HPC resource is much larger and consists of the Expanse HPC, a NSF XSEDE resource operated and managed by San Diego Supercomputer Center (SDSC) (https://www.sdsc.edu/services/hpc/expanse/expanse_architecture.html). In theory, the CyberGIS-Compute Service has the ability to support other HPCs as well. Users who do not wish to use HPC computational resources can use CJW VM computational resources directly to run the models. The CJW VM and HPC HydroShare resources can be found at Choi et al. (2021c) and Choi et al. (2021b), respectively.

It is necessary to compare the hardware specifications of the CJW VM and the Expanse HPC (Table 4.2). The CJW VM has only 3 compute nodes (i.e., machines) each of which has six CPUs with 2.50 GHz Clock Speed and 6 GB DRAM. Each user can only use one compute node and up to 6 CPUs and the nodes can be shared among users (according to the CJW setup at the time of writing of this research paper, if the browser is idle for 24 hours, the session gets expired). This means the maximum degree of parallelism for simulations using this computational resource can be only 6. Thus, in case of running one basin from the VB study (704 runs) and using all the 6 available CPUs, each CPU will need to run 117.33 simulations (some of them 117 and some other 118 times). The Expanse HPC has 728 AMD Rome standard compute nodes each of which is equipped with 256 GB DRAM and 128 2.25 GHz CPUs. The Expanse HPC allows the user to only use up to 2 nodes at a time, i.e., 256 CPUs or the maximum degree of parallelism for simulations. Thus, in case of running one basin from the VB study (704 runs) and using all the available 256 CPUs, each CPU will need to run 2.75 simulations (some of them 2 and some 3). We only use the Expanse HPC, because it is more powerful than the Keeling HPC. We recommend the CJW VM HydroShare resource only be used to run 1.5 years of simulation, 0.5 years of initialization plus one year and for one basin; this does not reproduce the methodology used in the

VB study but runs in about two hours. Therefore, the CJW VM resource may mainly be used for smaller simulations (a basin or two with a short period of simulation) and serve as an instructional tool. A debug mode can be used in the CJW VM HydroShare resource that runs only one day of simulation for testing purposes. On the other hand, the HPC HydroShare resource, when using the highest achievable number of CPUs for parallel simulations, runs six years of simulation, one year of initialization plus 5 years, as in the VB study, for four basins only in about 40 minutes. This shows how the HPC resource can scale up the model runs offering a high-performance tool. More details about the run-time performance of the notebooks are discussed in the results and discussion section.

Table 4.2 Hardware specifications of the computational environments

Computational Environment	Node count	Number of CPU cores per node (for parallel runs only)	Clock Speed (GHz)	DRAM/node (GB)
CJW VM*	3	6	2.50	6
Expanse HPC**	728	128	2.25	256

*Intel(R) Xeon(R) CPU E5-2680 v3. Each user can only use 1 node (up to 6 CPUs) and the nodes can be shared among users.
**AMD Rome Standard Compute Nodes. Each user can only use up to 2 nodes, which means 256 CPUs, the maximum number of parallelism for simulations.

The following subsections discuss the general purpose of each notebook. For specific coding details, refer to the notebooks in the HydroShare resources at Choi et al. (2021c) and Choi et al. (2021b).

4.2.3.1 Data processing notebook

The first notebook (JN 1: Preprocessing) processes the original CAMELS SUMMA files and the input forcing datasets (Table A2 in the Appendix). The user can select one or several CAMELS basins (1-671 basins) but by selecting a higher number of basins the computational time and expense increases. Notebook 1 subsets the original CAMELS SUMMA files, producing SUMMA attributes, parameters, initial conditions, and hourly NLDAS forcing files for the selected basin(s). Then, additional forcing datasets for the hydrologic model sensitivity study are developed from the NLDAS data files (FORCINGS box in Figure A1 in the Appendix) as discussed below.

For each SUMMA-model setup, variations in 14 SUMMA-generated outputs, described in Table A1 (in the Appendix), are examined with respect to variations in seven input forcings (air pressure (*prs*), air temperature (*tmp*), long wave radiation (*lwr*), precipitation rate (*ppt*), specific humidity (*hum*), shortwave radiation (*swr*), and wind speed (*wnd*)), under different model parameterizations and configurations. The SUMMA outputs generated with the one-hour NLDAS forcing dataset are considered the benchmark (NLDAS dataset 1; FORCINGS box in Figure A1 in the Appendix). The rest of datasets (*ppt* to *prs* datasets; FORCINGS box in Figure A1 in the

Appendix) are developed, holding each of the individual forcing variables constant over a 24-hour period while the other six forcing variables contain the original hourly NLDAS values.

Figure A2 in the Appendix shows the steps taken in the first notebook. As can be seen, this notebook is the same for the CJW VM and HPC environments except that the simulation time period and basins to be explored are pre-populated differently. The user still can change these setups in the third step of this notebook (step 1_3) as they wish. In the last step of this notebook, users can visualize the individual forcing variables held constant over a 24-hour period against the original hourly NLDAS values using hourly plots, and cumulative plots to see whether errors are compounding.

4.2.3.2 SUMMA execution notebook

The second notebook (JN 2: Running SUMMA) executes the SUMMA model using the input data from the first notebook for four different sets of SUMMA basin runs, outlined in Figure A1 in the Appendix (RUNS box) and described in detail in Van Beusekom et al. (2022). The first set of basin runs (DEFAULT; 8 SUMMA runs per basin; RUNS box) uses the eight forcing datasets (FORCINGS box) combined with default parameters and a default SUMMA configuration. The SUMMA default configuration is set in the resource model decision file.

The second set of basin runs (LHS; 88 SUMMA runs per basin; RUNS box) uses the eight forcing datasets combined with 11 parameter sets and a default SUMMA configuration. The 11 parameter sets consist of the default parameter set and 10 additional parameter sets with 15 commonly calibrated parameters (Table A2 in the Appendix). As detailed in Van Beusekom et al. (2022), the parameters are sampled using Latin Hypercube Sampling (LHS) over their defined range. The *pyDOE LHS* function was used (https://pythonhosted.org/pyDOE/randomized.html) to create unique 10 x 15 LHS sampling matrices for the selected basin. The LHS sets that are created using this notebook can be different from the ones used by the VB Study and that the choice of a different seed value will also lead to different LHS sets. The LHS matrices were used to produce 10 parameter files for the basin with random sampling of the 15 parameters while considering the parameter constraints listed in Table 4.2.

The third set of basin runs (CONFIG; 64 SUMMA runs per basin; RUNS box) uses the eight forcing datasets combined with the default parameter set and eight SUMMA configurations. The eight SUMMA configurations, outlined in the CONFIGURATIONS box in Figure A1 in the Appendix, test three model decisions (stomatal resistance (stomResist), choice of snow interception parameterization (snowIncept), and choice of canopy wind profile (windPrfile) with two options for each decision. Note the default configuration for this study is shown in bold in the CONFIGURATIONS box in Figure A1:BallBerry, lightSnow, and logBelowCanopy.

The fourth set of basin runs (COMPREHENSIVE; 704 SUMMA runs per basin; RUNS box in Figure A1 in the Appendix) includes the DEFAULT, LHS, and CONFIG basin runs, and is the only set that needs to be run to replicate a single basin sensitivity study following the VB study method (six years of simulation must be run for replication). For testing purposes, sets 1-3 can also be run by themselves. The 10 parameter set files for the basin from the LHS sampling plus the default parameters (11 parameter sets) are run each with eight SUMMA configurations (CONFIGURATIONS box in Figure A1 in the Appendix).

Figure A3 in the Appendix shows the steps taken in the second notebook. The first two steps in this notebook are the same for the CJW VM and HPC environments but the rest of the workflow differs. In the CJW VM notebook, the user can define the simulations, i.e., by selecting the simulation period or by selecting the model configuration or parameter values. Depending on which run complexity choice (i.e., DEFAULT, LHS, CONFIG, COMPREHENSIVE in the RUNS box in Figure A1 in the Appendix) is selected the notebook executes a specific set of code cells using a conditional statement logic (e.g., if user selects *config_prob == 1*, step 2_7 is run which leads to CONFIG runs as shown in the RUNS box in Figure A1 in the Appendix). As mentioned earlier in section 4.2.3, users need to carefully consider the number of basins and the length of the simulation period as the CJW VM environment is not powerful enough to run big problems in a reasonable time. In the HPC notebook, we only provided the user with the option to run the most complex problem, i.e., *lhs_config_prob*, as the HPC is powerful enough to run the full problem quickly making it unnecessary to allow for simpler problems. The user still can change the simulation period (in step 2-3 of the workflow in Figure A3). The other main difference between the CJW VM and HPC notebooks is that the codes calculating KGE values for the HPC notebook are moved over to the HPC and executed on the HPC (Step 2_8 in HPC branch in figure A3) while for the CJW VM environment, the KGE values are calculated locally on CJW VM (Step 2_9 in CJW VM branch in Figure A3). In the HPC environment, the KGE values are calculated on the HPC resource to prevent having to transfer large data volumes from the HPC to the CJW VM with the sole purpose of calculating performance metrics. Users can use Globus to transfer selected output files from HPC to the CJW VM for additional analysis. Notebook 4, which exists only in the HPC environment was developed for this purpose and is discussed in section 4.2.3.4.

A modified and scaled (range between -1 and 1) version of the KGE was used as an indicator of model output sensitivity to a change in input forcing based on the work of Clark et al. (2021) and Mathevet (2006) and is described in Van Beusekom et al. (2022). The KGE test compares hourly model outputs generated with the benchmark forcing dataset (NLDAS dataset 1; Table A2 in the Appendix) with outputs generated with the forcing datasets with one forcing held constant (CNST datasets 2-8; Table A2 in the Appendix). KGE values are ranked from low to high to determine relative order of forcing influence on model outputs with highest rankings associated with least influence of change to 24-hour constant forcing.

4.2.3.3 Post-processing notebook

The third notebook (JN 3: Post-processing) produces visualizations, utilizing the model outputs from Notebook 2, of the sensitivity of SUMMA model output to the temporal resolution of the model forcing. Figure A4 in the Appendix shows the steps taken in the third notebook. The notebooks for CJW VM and HPC environments are the same. For the selected basin(s), eight plots are generated with Notebook 3 that follow the analysis in Van Beusekom et al. (2022). The reader is referred to the supplementary materials and Van Beusekom et al. (2022) for a detailed explanation of each of the eight plots. In this paper, we only present the first two figures generated by Notebook 3, i.e., location of the selected CAMELS basin, and KGE values for each output variable for all 8 DEFAULT model runs.

4.2.3.4 Model output transfer

The fourth notebook (JN 4: Use Globus) is only included in the HPC HydroShare resource (Figure A5 in Appendix) to transfer SUMMA output files from HPC to CJW on HydroShare. To retrieve the data from the HPC, this notebook needs a job ID submitted to the HPC, which is created in Notebook 2. While this notebook is running, users can see the live status of the file transfer managed by the CyberGIS-Compute Service. Once running of this notebook is successfully finished, the user will be able to see the location of the transferred file on CJW.

4.2.4 Performance analysis

We tested the performance of the cyberinfrastructure using a number of model scenarios, varying the number of simulation years and basins for each computational environment, described in Table 4.3. For the CJW VM environment, we tested the performance of notebooks 1-3 for three scenarios (Table 4.3, rows 1 - 3): (1) four basins and 1.5 years of simulations (a total of six years of simulations), (2), and (3) one basin (scenario 2 and 3 basins are not the same) and six years of simulation (a total of six years of simulations). We decided not to test for more simulation years as the CJW VM notebook to run SUMMA simulations was slow and the HPC resource was available for larger simulations.

For the HPC environments, we used Expense HPC, and tested the performance of notebooks 1-3 for 12 scenarios (Table 4.3, rows 4 - 15). In these scenarios, we varied the number of allocated CPUs (128 or 256) for parallelism and the total number of simulations years ranging from one basin and six years of simulation (a total of six years of simulations) to 20 basins and six years of simulation (a total of 120 years of simulations, which equals about three percent of the total simulation years for the whole VB Study). To test the performance of Notebook 4, transferring output files from HPC to the CJW, we only used scenarios HPC_256_1 to HPC_ 256_6 (rows 4 - 9 in Table 4.3) and repeated each transfer 5 times to obtain a range of run-time for each of the scenarios.

Table 4.3 Model scenarios for notebooks run-time performance analysis

Row	Model scenario name	Number of CPU cores allocated	Number of basins	Simulation years	Total simulation years
1	CJWVM_1	6	4	1.5	6
2	CJWVM_2	6	1	6	6
3	CJWVM_3	6	1	6	6
4	HPC_256_1	256	1	6	6
5	HPC_256_2	256	4	6	24
6	HPC_256_3	256	6	6	36
7	HPC_256_4	256	10	6	60
8	HPC_256_5	256	15	6	90
9	HPC_256_6	256	20	6	120
10	HPC_128_1	128	1	6	6
11	HPC_128_2	128	4	6	24
12	HPC_128_3	128	6	6	36
13	HPC_128_4	128	10	6	60
14	HPC_128_5	128	15	6	90
15	HPC_128_6	128	20	6	120

4.3 Results and discussion

In this section, we first present results of the modeling case study that served as a motivation for the cyberinfrastructure. Then, we present results of a performance analysis focusing on contrasting the HPC and CJW VM notebooks using a variety of model setups. Then, we summarize the resulting resources from this study that are shared on HydroShare. Finally, we discuss opportunities and challenges identified through this research that can be the focus of future research.

4.3.1 Results of the modeling case study

The four CAMELS basins with diverse basin attributes (Table 4.4) and a variety of forcing influences were chosen as individual basin examples below. The diverse attributes, as shown in Table 4.4 for these four basins, make them ideal to show how this diversity affects the modeling results. For inter-basin comparison purposes, Figure 4.3 shows the KGE values using the DEFAULT model runs for each CNST dataset (datasets 2-8; Table A2 in the Appendix), grouped by SUMMA output variable.

Table 4.4. Station descriptions for basin individual analysis

USGS Station ID	**Name**	**CAMELS Attributes**						
		Drainage area (km^2)	Gage datum (m)	Mean daily precipitation (mm/day)	Fraction of precipitation falling as snow	Aridity	Mean daily discharge (mm/day)	Runoff ratio*
01632900	Smith Creek Near New Market, VA	242	268	2.91	0.10	0.89	0.80	0.27
02212600	Falling Creek near Juliette, GA	187	1202	3.37	0.01	1.19	0.74	0.22
09378630	Recapture Creek Near Blanding, UT	10	2195	1.58	0.50	0.50	0.21	0.13
11264500	Merced River at Happy Isles Bridge near Yosemite, CA	469	1228	2.64	0.91	1.15	1.94	0.73

* Annual runoff / annual precipitation

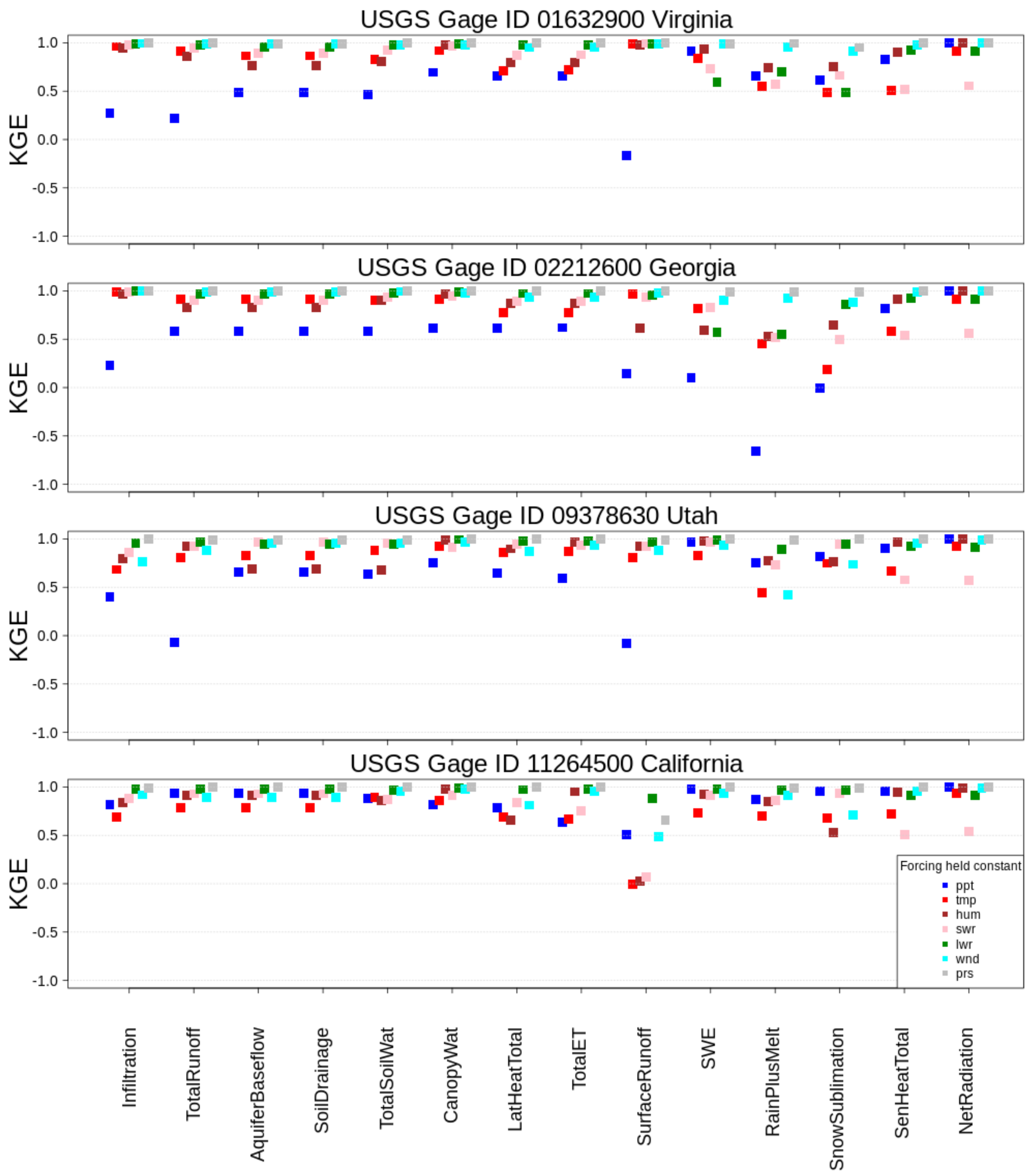


Figure 4.3. KGE values using the DEFAULT model runs for each CNST dataset (datasets 2-8; Table A2 in the Appendix), grouped by SUMMA output variable

The example shown in Figure 4.3 uses the DEFAULT simulations (Section 4.2.3.2). The simulations consist of one reference simulation in which all forcing variables vary on an hourly basis (NLDAS dataset 1; FORCINGS box in Figure A1 in the Appendix) and seven simulations in which one forcing variable is held constant at the mean daily value throughout each day. (the seven datasets *ppt* to *prs*; FORCINGS box in Figure A1 in the Appendix). KGE values were calculated relative to the reference simulation for each of the seven simulations using five years of hourly model output from 10/1/1991 - 9/30/1996. For the four selected basins, Figure 4.3 shows the KGE values for each SUMMA output variable using the DEFAULT model runs. In Figure 4.3, the first three gages (01632900, 02212600, and 09378630) show a strong *ppt* temporal aggregation influence using DEFAULT, whereas gage 11264500 is more influenced by *tmp*, *hum*, and *swr* temporal aggregation. In other words, it is crucial to have a higher temporal resolution for the aforementioned forcing variables in the given basins and DEFAULT run to reduce the associated hydrologic model uncertainties. The weaker influence of *ppt* temporal aggregation on the gage 11264500 compared to other gages can be attributed to its high value of fraction of precipitation falling as snow, 0.91 as opposed to 0.1, 0.01, 0.5 (Table 4.4). Also in Figure 4.3, we see varying ranges in KGE values for particular output variables. As an example, SurfaceRunoff is affected by constant hourly values of: *ppt* for gages 01632900 and 09378630; *ppt* and *hum* for gage 02212600; and *tmp*, *hum*, *swr*, *wnd*, *ppt*, and *prs* (most to least dominant) for gage 11264500. This shows the forcing variables in each basin that need to have a higher temporal resolution to reduce the uncertainties in the SurfaceRunoff output. All in all, the information contained in Figure 4.3 demonstrates the variability in model output sensitivity to the temporal resolution of the forcing variables in particular when comparing gage 11264500 to other three gates. In this section, we only presented one figure to make an inter-basin comparison to show how different the results can be across different basins. Researchers can further explore the differences between individual basins using other plots that can be made using the interactive IPython notebooks. The notebooks reproduce the results from the original VB study.

4.3.2 Results from performance analysis

The run-time for the data processing notebook (Notebook 1) and the post-processing notebook (Notebook 3), the two notebooks that are very similar between CJW VM and HPC computational environments, for the 15 scenarios in Table 4.3 are presented in Figure 4.4. In this figure, some data points fully overlap, for example, for the scenarios CJWVM_2, HPC_256_1 and HPC_256_2 the run-time for the Notebook 1 was one minute and 36 seconds. As expected Notebooks 1 and 3 do not take a significant time to run because they are only preprocessing and output analysis notebook which do not run the simulations. For scenarios with fewer than 30 simulation years, Notebook 1 takes longer than Notebook 3, but this changes for scenarios with more simulation years. For example, Notebook 3 requires 3.3 times longer than Notebook 1 for scenarios HPC_256_6 or HPC_128_6 with 120 simulation years. For the CJW VM environment, the average time to run Notebooks 1 and 3 across the tested scenarios only takes 0.6% of the entire time needed to run all Notebooks 1, 2, and 3. This means the time required to run data processing and post-processing notebooks is not a limiting factor for running the simulations. For the HPC environment, this ratio increases to 8.5% and 11.3% when using 128 and 256 CPUs, respectively. This dramatic increase in the ratio is due to the significant decrease in run-time of Notebook 2 when using HPC which is discussed next.

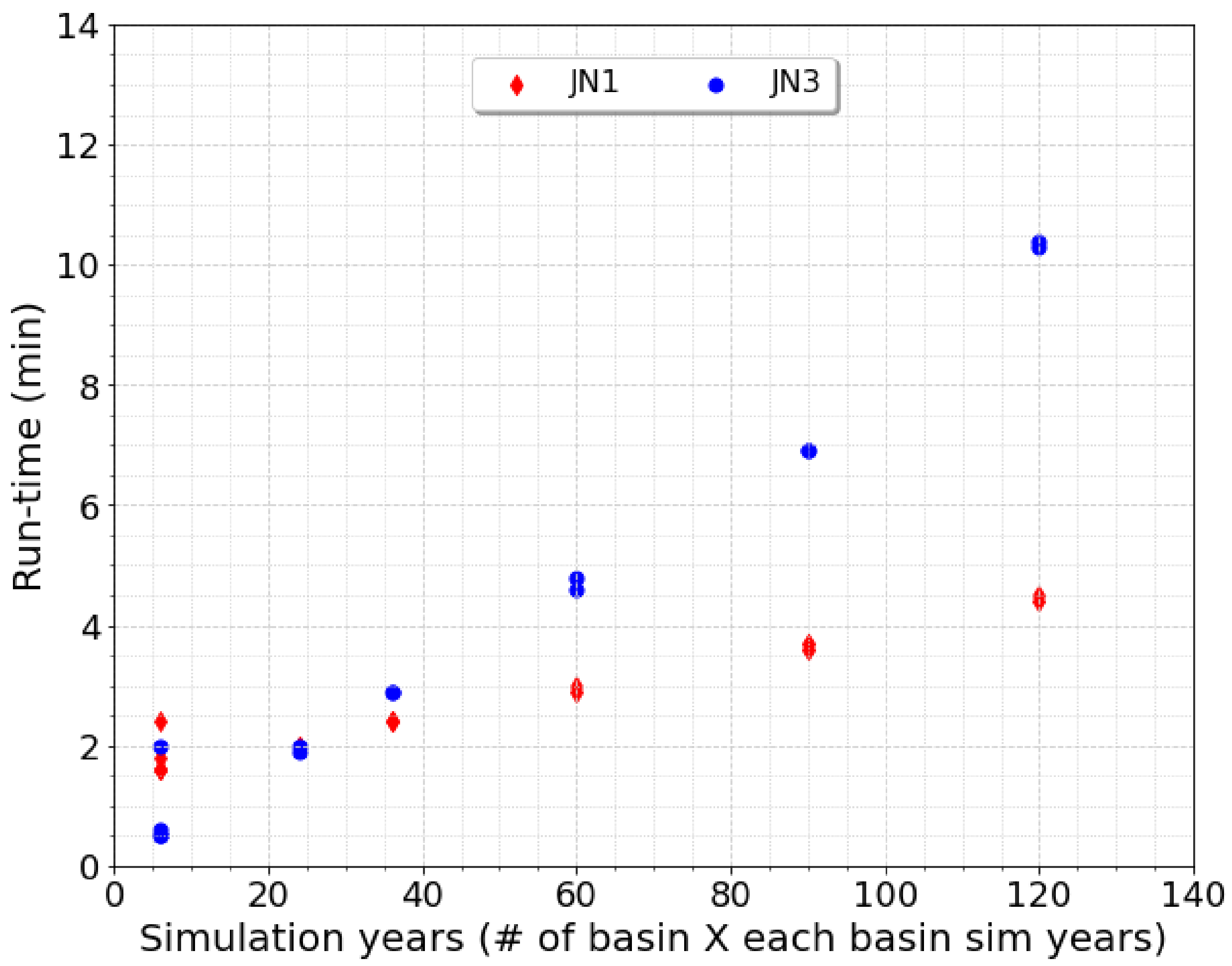


Figure 4.4. Notebook 1 and 3 run-time performance analysis for different model simulations (both notebooks 1 and 3 were run on CJW VM no matter whether the HPC or CJW VM environment was used for the modeling; therefore we do not distinguish between the environments

The run-time for the SUMMA execution notebook (Notebook 2) for the 15 model scenarios using different computation environments is shown in Figure 4.5. The CJW VM environment took between 7 hr 33 min to 9 hr 25 min to run Notebook 2 for only six years of simulations. This emphasizes that, while the CJW VM environment is technically able to simulate smaller models, it is not efficient to run larger simulations. The HPC, however, performed the simulations much faster, as expected. In the case of running one basin for six years, the HPC was 42 and 36 times faster than the CJW VM, when using 256 and 128 CPUs, respectively. HPC with 256 CPUs (scenario HPC_256_6) could finish the simulations for 120 years (3% percent of the VB study) in only 2 hr and 10 min while HPC with 128 CPUs (scenario HPC_128_6) could run the same problem in 3 hr and 12 min (1.48 times of the time need by HPC_256_6). Using the HPC with 256 CPUs, assuming a linear extrapolation, the SUMMA simulations from Notebook 2 are expected to be done in about 75 hours for the entire VB Study. All in all, it can be easily concluded that

HPCs provide significantly faster simulations making them ideal to run for larger studies. When using the HPC resource and in the case of 120 years of simulation, dividing the number of the allocated CPUs by two increased the run-time only by about 50% and not 100% as one might expect. This non-linear scaling can be mainly attributed to different factors including 1) communication overhead in the computational resource that reduces scaling, and 2) the fact that some parts of the codes in Notebook 2 did not utilize parallelism. For example, KGE values were only calculated after they were exported as NetCDF files instead of being calculated directly from the raw SUMMA output files.

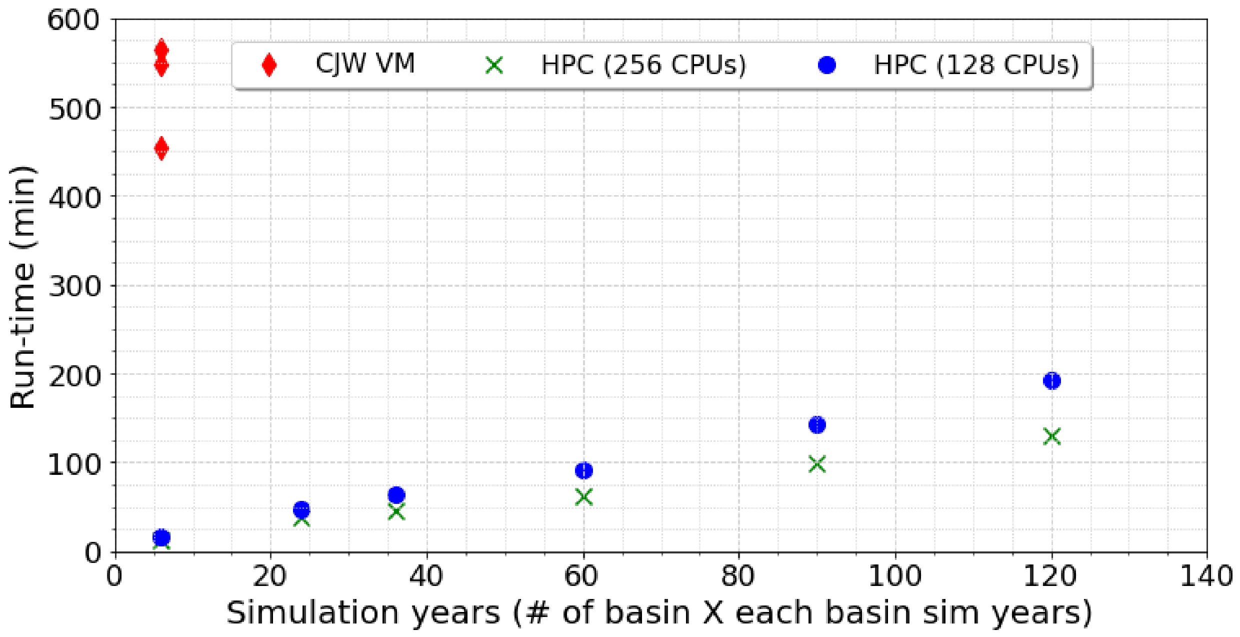


Figure 4.5. Notebook 2 run-time performance analysis for different model simulations using the CJW VM, or HPC (Expanse with 256 or 128 CPUs) options

The run-time for transferring the SUMMA output files from Expanse HPC to CJW on HydroShare using the Globus service integrated by CyberGIS-Compute Service is shown in Figure 4.6. Each transfer was repeated 5 times to obtain a range of run-time for each of the model simulations with a different total number of simulation years. Figure 4.6 shows that the range of the transfer time for each total number of simulation years is small indicating a consistent data transfer. For 120 years of simulation, by average it took about 14.5 min to transfer about 118 GB of data from HPC to CJW which highlights that the approach used to transfer data from HPC to CJW is fast and stable. Moreover, a linear relationship between the size of the data transferred (or number of total simulation years and the run-time (averaged over the five repetitions) can be easily seen.

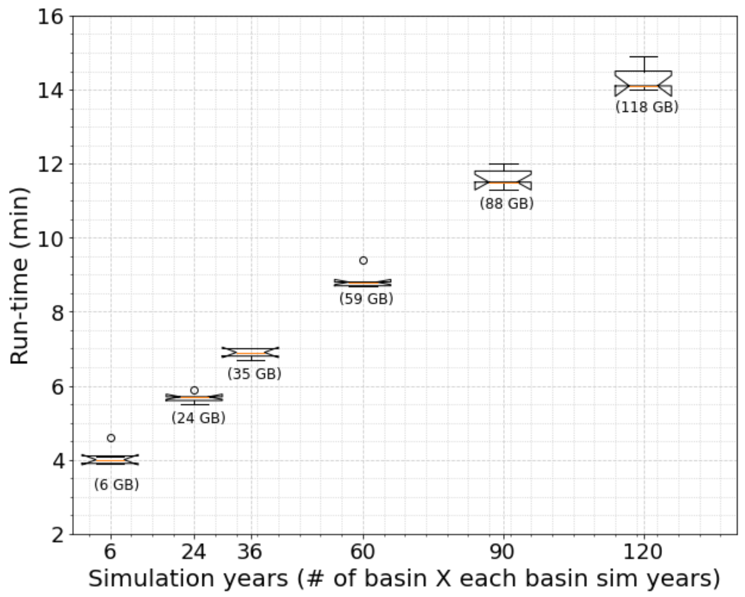
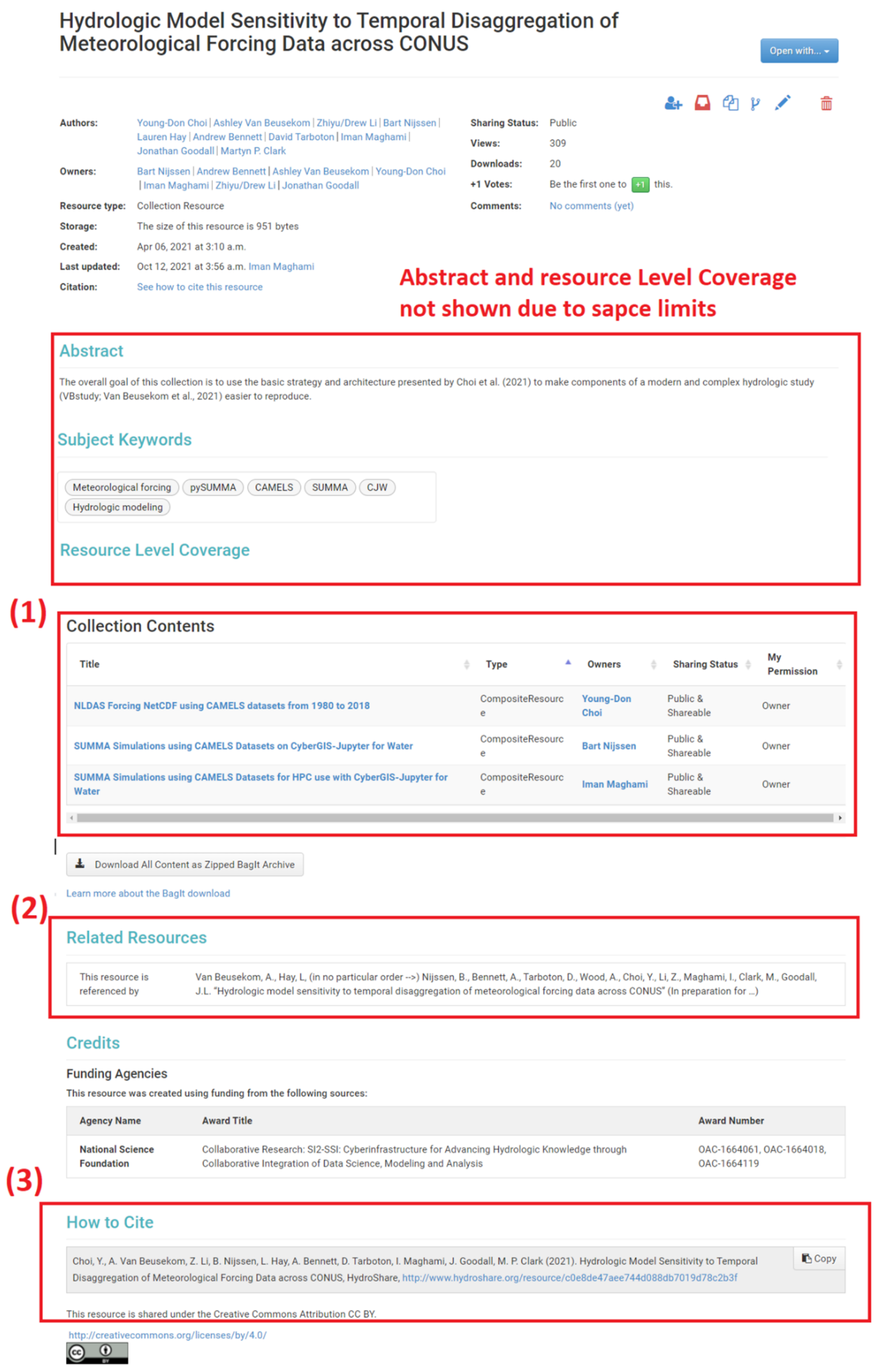


Figure 4.6. Boxplots for Notebook 4 run-time performance analysis for six different simulation years to transfer data from Expanse HPC to CJW on HydroShare. Each transfer was repeated five times to obtain a range of run-time for each of the model simulations

4.3.3 Best-practice for reproducibility

The cyberinfrastructure presented in this paper made it possible to reproduce a complex research study, providing interactive codes that can easily be used to test the theories put forth by the original study. This kind of conclusion-reproducibility is a higher level of transparency in modeling studies than the usual bit-reproducibility, which often provides code that is not of practical use to another researcher but is simply runnable. The data for this study was pre-processed and the output post-processed by using existing software in Python packages. The study here demonstrated best-practice by using the online repository of HydroShare to not only store data and modeling code, but to also store computational environment, API version documentation and container installation. HydroShare, as a hydrology-based repository service, facilitated this by allowing all the parts of the problem to be stored together as one resource. Furthermore, parts of the resource can be pulled out and made into a new version (updated, revised, or modified), in order to promote collaboration.

To this point, a HydroShare collection resource was created that contained three composite resources. These resources are published and have Digital Object Identifier (DOI) which makes them immutable and findable. Figure 4.7 shows the landing page for the HydroShare collection resource. The three composite resources that are contained by this resource are shown in dialogue box 1, the “Related Resources” in box 2 refers to this paper, and box 3 shows the information on how to cite this resource. Figure 4.8 shows the landing page for the HydroShare composite resource holding the HPC notebooks. Box 1 shows the contents of the resource, most importantly the four Jupyter notebooks, and the readme.md file. The readme.md (box 2) provides the user with the instruction on how to run the notebooks. Box 3 shows the information on how to cite this HydroShare resource.



Hydrologic Model Sensitivity to Temporal Disaggregation of Meteorological Forcing Data across CONUS

Authors:	Young-Don Choi | Ashley Van Beusekom | Zhiyu/Drew Li | Bart Nijssen | Lauren Hay | Andrew Bennett | David Tarboton | Iman Maghami | Jonathan Goodall | Martyn P. Clark
Owners:	Bart Nijssen | Andrew Bennett | Ashley Van Beusekom | Young-Don Choi | Iman Maghami | Zhiyu/Drew Li | Jonathan Goodall
Resource type:	Collection Resource
Storage:	The size of this resource is 951 bytes
Created:	Apr 06, 2021 at 3:10 a.m.
Last updated:	Oct 12, 2021 at 3:56 a.m. Iman Maghami
Citation:	See how to cite this resource

Sharing Status:	Public
Views:	309
Downloads:	20
+1 Votes:	Be the first one to +1 this.
Comments:	No comments (yet)

Abstract and resource Level Coverage not shown due to sapce limits

Abstract

The overall goal of this collection is to use the basic strategy and architecture presented by Choi et al. (2021) to make components of a modern and complex hydrologic study (VBstudy; Van Beusekom et al., 2021) easier to reproduce.

Subject Keywords

Meteorological forcing · pySUMMA · CAMELS · SUMMA · CJW · Hydrologic modeling

Resource Level Coverage

(1)

Collection Contents

Title	Type	Owners	Sharing Status	My Permission
NLDAS Forcing NetCDF using CAMELS datasets from 1980 to 2018	CompositeResource	Young-Don Choi	Public & Shareable	Owner
SUMMA Simulations using CAMELS Datasets on CyberGIS-Jupyter for Water	CompositeResource	Bart Nijssen	Public & Shareable	Owner
SUMMA Simulations using CAMELS Datasets for HPC use with CyberGIS-Jupyter for Water	CompositeResource	Iman Maghami	Public & Shareable	Owner

Learn more about the BagIt download

(2)

Related Resources

This resource is referenced by	Van Beusekom, A., Hay, L, (in no particular order -->) Nijssen, B., Bennett, A., Tarboton, D., Wood, A., Choi, Y., Li, Z., Maghami, I., Clark, M., Goodall, J.L. "Hydrologic model sensitivity to temporal disaggregation of meteorological forcing data across CONUS" (In preparation for ...)

Credits

Funding Agencies

This resource was created using funding from the following sources:

Agency Name	Award Title	Award Number
National Science Foundation	Collaborative Research: SI2-SSI: Cyberinfrastructure for Advancing Hydrologic Knowledge through Collaborative Integration of Data Science, Modeling and Analysis	OAC-1664061, OAC-1664018, OAC-1664119

(3)

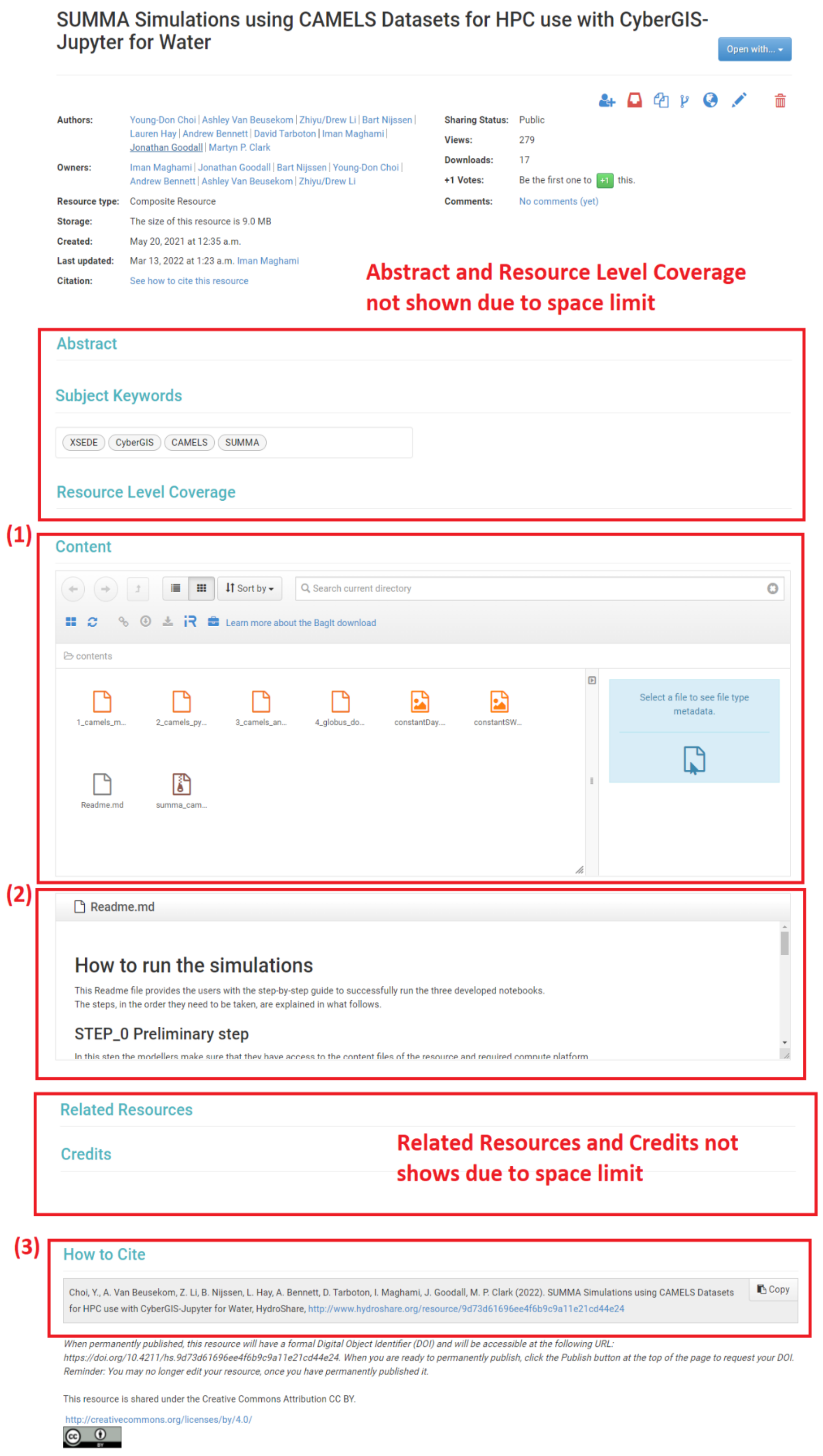
How to Cite

Choi, Y., A. Van Beusekom, Z. Li, B. Nijssen, L. Hay, A. Bennett, D. Tarboton, I. Maghami, J. Goodall, M. P. Clark (2021). Hydrologic Model Sensitivity to Temporal Disaggregation of Meteorological Forcing Data across CONUS, HydroShare, http://www.hydroshare.org/resource/c0e8de47aee744d088db7019d78c2b3f

This resource is shared under the Creative Commons Attribution CC BY.

http://creativecommons.org/licenses/by/4.0/

Figure 4.7. The HydroShare landing page for the collection resource developed by this study (2021a)



SUMMA Simulations using CAMELS Datasets for HPC use with CyberGIS-Jupyter for Water

Authors: Young-Don Choi | Ashley Van Beusekom | Zhiyu/Drew Li | Bart Nijssen | Lauren Hay | Andrew Bennett | David Tarboton | Iman Maghami | Jonathan Goodall | Martyn P. Clark

Owners: Iman Maghami | Jonathan Goodall | Bart Nijssen | Young-Don Choi | Andrew Bennett | Ashley Van Beusekom | Zhiyu/Drew Li

Resource type: Composite Resource

Storage: The size of this resource is 9.0 MB

Created: May 20, 2021 at 12:35 a.m.

Last updated: Mar 13, 2022 at 1:23 a.m. Iman Maghami

Citation: See how to cite this resource

Sharing Status: Public

Views: 279

Downloads: 17

+1 Votes: Be the first one to +1 this.

Comments: No comments (yet)

Abstract and Resource Level Coverage not shown due to space limit

Abstract

Subject Keywords

XSEDE CyberGIS CAMELS SUMMA

Resource Level Coverage

(1) Content

contents

1_camels_m... 2_camels_py... 3_camels_an... 4_globus_do... constantDay... constantSW... Readme.md summa_cam...

Select a file to see file type metadata.

(2) Readme.md

How to run the simulations

This Readme file provides the users with the step-by-step guide to successfully run the three developed notebooks.
The steps, in the order they need to be taken, are explained in what follows.

STEP_0 Preliminary step

In this step the modellers make sure that they have access to the content files of the resource and required compute platform

Related Resources

Credits

Related Resources and Credits not shows due to space limit

(3) How to Cite

Choi, Y., A. Van Beusekom, Z. Li, B. Nijssen, L. Hay, A. Bennett, D. Tarboton, I. Maghami, J. Goodall, M. P. Clark (2022). SUMMA Simulations using CAMELS Datasets for HPC use with CyberGIS-Jupyter for Water, HydroShare, http://www.hydroshare.org/resource/9d73d61696ee4f6b9c9a11e21cd44e24

When permanently published, this resource will have a formal Digital Object Identifier (DOI) and will be accessible at the following URL: https://doi.org/10.4211/hs.9d73d61696ee4f6b9c9a11e21cd44e24. When you are ready to permanently publish, click the Publish button at the top of the page to request your DOI. Reminder: You may no longer edit your resource, once you have permanently published it.

This resource is shared under the Creative Commons Attribution CC BY.

http://creativecommons.org/licenses/by/4.0/

Figure 4.8. The HydroShare landing page for the HPC resource developed by this study (2021b)

4.3.4 Opportunities and challenges

The efforts of the conclusion-reproducibility setup presented here can be used to test different situations where the theories proposed in the VB study and in the individual basins explored in section 4.3.1 can be confirmed or even new patterns to be detected. With minimal changes to the notebooks, a user could (1) use different CAMELS basins, (2) use different parameters in the LHS set, (3) use different time periods, e.g. a drought period, (4) use more than 10 LHS sets, e.g., a more thorough exploration of the parameter space, and (5) use different or more model configuration, combining different or more model decisions.

The last two changes, i.e., using a higher number of LHS sets, and using different model configuration/decisions, bring up the major first challenge that was encountered in reproducing this big-data problem; it is a big-data problem. Most conclusion-reproducibility setups are used with toy-problems. Here, the limit on manageable data size was pushed, even with only running a few basins. HPC computational power was required to run the full 6 years of simulation, and expanding the parameter exploration space or adding model decisions would compound the size.

The second major challenge that is encountered is implementing version control. What if users need to run the Jupyter notebooks presented in this study in their own computational environment (not deployed on CJW), or they need to install a newer version of a model API? How can they make sure they have a reproducible framework that is robust enough to tackle the version control problem? Here, we propose a way for future research to tackle the version control by making the computational environment all documented and installable via a Python environment file. This means the hydrology modeling code, pySUMMA, has to be installable by Anaconda. In the future, Python packages updates will break compatibility, but compatibility can be preserved by installing the older versions (as documented in the environment file), or the user understanding the updates in order to manually work around the updated package incompatibility. This would be expected to mostly be an issue with updates of pySUMMA, since it is doing the heavy lifting and the pre-processing and post-processing Python packages would not be expected to change as much. If a researcher wants to use a newer (future) version of pySUMMA, they may need to debug some parts of the Jupyter notebooks that are affected by the changes. While this is not an ideal way to handle version updates, at least the researcher has options of a working, albeit older, computational environment, from which to begin reproducing the study.

The specifics of the environment can be placed in a Python environment.yml file that can be shared as part of the online model and data repositories, and can be installed with an installation notebook inside the repository. This can use best practice for transparency about what dependencies the computational gateway interface notebooks need to run. The specifics of each dependency can be described in the installation notebook, so that if in the future there are issues with the availability of that dependency, then a suitable substitute can be found. Version control issues can be thus addressed through this methodology, albeit an imperfect solution depending on possible user troubleshooting.

In addition to the major challenges explored above, there are two more items: 1) transferring data from CJW on HydroShare to the user’s local machine, and 2) what if a user wants/needs to set up their own HPC to reproduce this study or for similar use cases.

For cases when users need to transfer data from CJW on HydroShare to their local machine (e.g., they need to permanently store all the model outputs), if the data size is small it is easily doable. However, for big data problems similar to the entire VB study or even the four selected basins study explored in this paper, this transfer would be slow or unstable. In fact, if a user needs to have the big output on a local computer, it should be downloaded from HPC directly to the user's local computer using Globus (without going through CJW). This would need the user's computer to become a Globus server. This can be considered as a limitation of this work and a topic of future research.

Finally, if users need to set up their own HPC to work with similar Jupyter notebooks, they need to set up their own job submission service and configure it to their own account from the HPC that they have access to. Although the job submission software used in this study is open source, it is customized for the UIUC HPC so it cannot be directly used for this purpose. One future work can be that CJW asks users to provide their own credentials and to their own HPC, rather than using the UIUC HPC service.

4.4 Conclusions

The importance of scientific research reproducibility is broadly recognized across different scientific disciplines. When it comes to computational hydrology, reproducing large-scale studies, being computationally expensive and time-consuming, can be significantly challenging. This research showed how using a basic strategy and architecture integrating the (1) online data repositories, (2) computational environments, and (3) model API presented by Choi et al. (2021d) could facilitate reproduction of the components of modern and complex hydrologic studies (first research objective). For this purpose, we used a recently published large-scale hydrologic research (VB study) as an example. We designed and developed cyberinfrastructure that utilized software components to enable intuitive, and online access to computational environments. This approach was used to remove the potential software inconsistencies from users' differing personal software editions, as well as to make implementation easier with pre-compiled software, with the added complication of a computationally expensive research problem instead of a case study. This approach gave the user the option to use either the CJW VM or HPC computational environments depending on how much they need to reproduce a problem more representative of the big-data problem. Using HydroShare as the data repository, and containerization of the pySUMMA API (with Docker or Singularity in the case of the HPC environment) along with a computational gateway interface of Jupyter IPython notebooks both hosted on the CJW made this possible. Three Jupyter notebooks for the CJW VM environment and four Jupyter notebooks for HPC environment were developed. Notebooks 1-3 for both CJW VM and HPC environments enable, (1) preparing the forcing data, simulation period, and study CAMELS basins, (2) executing SUMMA hydrologic model, and (3) visualization of the results. Notebook 4, only developed for the HPC environment, enables transferring large data from HPC to the scientific cloud service (i.e., CJW) using Globus service integrated by CyberGIS-Compute in a reliable, high-performance and fast way.

We analyzed performance of the notebooks focusing on contrasting HPC and CJW VM notebooks using a variety of model scenarios (second research objective). It was clear how the

HPC environments could perform significantly faster simulations compared to CJW VM, enabling users to explore a large number of basins and simulation periods. This clearly showed how the use of HPC could advance the reproducibility of modern and complex hydrologic studies. The run-time performance analysis for the big data transfer notebook for the HPC environment showed that the method used was stable, reliable and fast. Therefore, similar studies could easily benefit from the same approach for transferring large data between scientific cloud services.

With the focus of this research on the conclusion-reproducibility over the bit-reproducibility of the VB study, it was discussed that users can easily modify the notebooks to test different situations by varying the study basins and periods, parameterizations, and model configurations. These variations were shown that can lead to two major challenges (third research objective). First, adding up to the complexity of the big-data problem which was shown how it could be tackled by using the HPC computation environment. Second, the challenge of implementation of a version control system, e.g., when a user needs to install a newer version of a model API or when a user needs to run these codes on their local machine rather than the used cloud-based computational environment, was discussed. Sharing the dependencies of the computational environments as a Python environment yml file and an installation notebook that installs them was proposed as a future solution to tackle the version control issue despite the fact that it is an imperfect solution depending on possible user troubleshooting.

Finally, as a broader impact, the VB study methodology replicated with interactive codes could also serve as a valuable educational resource, allowing educators to present sophisticated modeling experiments for use within classrooms through online Python notebooks. Likewise, the basic approach could be extended to enable new water decision-support systems that take advantage of the SUMMA framework and HPC yet remain easy to interact with through notebooks to, for example, evaluate forcing sensitivity to a water resources management objective in a systematic manner.

Acknowledgments

This work was supported by the National Science Foundation under collaborative grants 1664061, 1664119 and 1664018 for the development of HydroShare (http://www.hydroshare.org). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Software and Data Availability

The data and Jupyter notebooks used in this study were shared on HydroShare with persistent DOIs. A collection resource in Hydroshare (Choi et al., 2021a) holds the resources containing the data and Jupyter notebooks further described in the following table. A Hydroshare account (http://hydroshare.org) and access to the CyberGIS-Jupyter for Water platform (accessed through HydroShare) are required to execute the Jupyter notebooks in the second and third resources.

Resource Description	Reference
Original NLDAS forcings for the CAMELS basins can be obtained as a NetCDF file*	Mizukami and Wood (2021)
SUMMA Simulations using CAMELS Datasets on CyberGIS-Jupyter for Water**	Choi et al. (2021c)
SUMMA Simulations using CAMELS Datasets for HPC use with CyberGIS-Jupyter for Water**	Choi et al. (2021b)

*The data from the CAMELS dataset (Newman et al., 2015a) was consolidated into one NetCDF file taking advantage of OPeNDAP data services (Tarboton and Calloway, 2021)

**The SUMMA setup for the CAMELS basins can be obtained from the summa_camels folder of the HydroShare resources.

List of relevant URLs

CyberGIS-Jupyter for Water: https://go.illinois.edu//cybergis-jupyter-water
Docker: https://www.docker.com
HydroShare REST API: https://github.com/hydroshare/hydroshare/wiki/HydroShare-REST-API
Numpy: https://www.numpy.org
Pandas: https://pandas.pydata.org
pySUMMA: https://github.com/UW-Hydro/pysumma/releases/tag/v3.0.3
Seaborn: https://seaborn.pydata.org
Singularity: https://sylabs.io
SUMMA: https://github.com/CH-Earth/summa/releases/tag/v3.0.3
xarray: http://xarray.pydata.org
XSEDE: https://www.xsede.org

Chapter 5

Conclusions

The overall objective of this dissertation research was to create methods for advancing the reproducibility of environmental, including hydrological, modelling. As stated in Chapter 1, the major goals of this research were: (1) to introduce an extensible environmental model metadata framework using a user-defined metadata schema, (2) to design a model instance metadata schema for a specific environmental model, and (3) to demonstrate how using a basic strategy and architecture integrating the online data repositories, computational environments, and model API presented by Choi et al. (2021d) could facilitate reproduction of the components of modern and complex hydrologic studies.

A flexible and scalable metadata framework for hydrologic models was introduced using a novel, user-defined model instance metadata schema. This approach transfers the capability of providing and maintaining metadata schemas for any arbitrary model program from the web-based repository developers to the modeling community. Because it is standardized and machine-readable, the schema can be consumed and utilized in a scalable way. The described model metadata framework is implemented in the web-based repository, HydroShare, to show how support can be provided for an indefinite variety of environmental and hydrologic models with little cost to the repository developers. A SWAT model was used as an example to describe the metadata schema specification.

This dissertation introduced a design process for creating a model instance metadata schema with an example of following this process for a specific hydrologic model: SUMMA. The proposed approach assists in a flexible and scalable model metadata framework. Upon creation of a number of different metadata schemas for different models by modelers, there will be a need to organize the metadata schemas so that they can be easily shared, found, and reused. Therefore, a method for organizing the metadata schemas is presented to advance the reportability and transparency.

Finally, the research in this dissertation, presents cyberinfrastructure for improving the reproducibility of complex, big-data hydrologic studies. To achieve this, the basic strategy and architecture presented by Choi et al. (2021d) was applied to a large-scale hydrologic study described in Van Beusekom et al. (2022). In this dissertation, cyberinfrastructure that utilized software components to enable intuitive and online access to computational environments was designed and developed. Through the lessons learned from this application, the challenges introduced by data and computational intensive modeling studies were identified and addressed. Most importantly, users were given the option to choose between non-HPC and HPC computational environments depending on if they need to reproduce a problem more representative of the big-data problem. Model APIs and Jupyter notebooks allow for ease of use and documentation of modeling experiments.

Across these three studies, the major contributions of the research include the following. (1) A flexible and scalable model agnostic metadata framework was developed which specifies the metadata elements for the model instances of any model program using a machine-readable model metadata schema file. (2) A methodology for building a model instance metadata schema for model programs was developed and strategies for how to manage the variety of metadata schemas that might be developed by the modeling community was discussed. (3) Finally, the cyberinfrastructure required to overcome data management and software architecture challenges to reproduce and reuse large-scale computational hydrological studies was advanced.

The approaches presented in this dissertation not only impact the discussed hydrologic and environmental fields, but also are general so that they can be adopted by other scientific disciplines that follow a similar practice of making use of model programs and model instances in their research and relying on similar cyberinfrastructure to reproduce and reuse large and complex studies. Ultimately, these approaches seek to foster a culture improving reproducibility and replicability among the model-based studies through facilitating the sharing, discovery, and reuse of data and models built by others.

References

Addor, N., Melsen, L.A., 2019. Legacy, Rather Than Adequacy, Drives the Selection of Hydrological Models. Water Resour. Res. 55, 378–390. https://doi.org/10.1029/2018WR022958

Baker, M., 2016. Is there a reproducibility crisis? Nature 533, 452–454. https://doi.org/10.1038/533452a

Chard, K., Tuecke, S., Foster, I., 2016. Globus: Recent enhancements and future plans, in: Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale. https://doi.org/10.1145/2949550.2949554

Chen, M., Voinov, A., Ames, D.P., Kettner, A.J., Goodall, J., Jakeman, A.J., Barton, M.C., Harpham, Q., Cuddy, S.M., DeLuca, C., Yue, S., Wang, J., Zhang, F., Wen, Y., Lü, G., 2020. Position paper: Open web-distributed integrated geographic modelling and simulation to enable broader participation and applications. Earth-Science Rev. https://doi.org/10.1016/j.earscirev.2020.103223

Choi, Y.-D., 2022. Migrated: Toward Open and Reproducible Environmental Modeling by Integrating Online Data Repositories, Computational Environments, and Model Application Programming Interfaces [WWW Document]. URL http://www.hydroshare.org/resource/9efc88837a534ebf91ea4e299cba5604

Choi, Y.-D., 2020a. Toward Open and Reproducible Environmental Modeling by Integrating Online Data Repositories, Computational Environments, and Model Application Programming Interfaces [WWW Document]. URL https://www.hydroshare.org/resource/33cfb9622a354442b2b0a869b15ea6b0/

Choi, Y.-D., 2020b. The Impact of Stomatal Resistance Parameterizations on ET of SUMMA Model in Aspen stand at Reynolds Mountain East [WWW Document]. HydroShare. URL https://www.hydroshare.org/resource/13d6b84a9553410297a67fa366a56cb2/ (accessed 10.7.22).

Choi, Y.-D., Beusekom, A. Van, Li, Z., Nijssen, B., Hay, L., Bennett, A., Tarboton, D., Maghami, I., Goodall, J., Clark, M.P., 2021a. Hydrologic Model Sensitivity to Temporal Disaggregation of Meteorological Forcing Data across CONUS [WWW Document]. HydroShare. URL https://www.hydroshare.org/resource/c0e8de47aee744d088db7019d78c2b3f/

Choi, Y.-D., Beusekom, A. Van, Li, Z., Nijssen, B., Hay, L., Bennett, A., Tarboton, D., Maghami, I., Goodall, J., Clark, M.P., 2021b. SUMMA Simulations using CAMELS Datasets for HPC use with CyberGIS-Jupyter for Water [WWW Document]. HydroShare. URL https://www.hydroshare.org/resource/9d73d61696ee4f6b9c9a11e21cd44e24/

Choi, Y.-D., Beusekom, A. Van, Li, Z., Nijssen, B., Hay, L., Bennett, A., Tarboton, D., Maghami, I., Goodall, J., Clark, M.P., 2021c. SUMMA Simulations using CAMELS Datasets on CyberGIS-Jupyter for Water [WWW Document]. HydroShare. URL https://www.hydroshare.org/resource/50e9716922dc487981b71e2e11f3bb5d/

Choi, Y.-D., Goodall, J., Sadler, J.M., Castronova, A.M., Bennett, A., Li, Z., Nijssen, B., Wang, S., Clark, M.P., Ames, D.P., Horsburgh, J.S., Yi, H., Bandaragoda, C., Seul, M., Hooper, R., Tarboton, D.G., 2021d. Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model Application Programming Interfaces. Environ. Model. Softw. 135. https://doi.org/10.1016/j.envsoft.2020.104888

Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J.E., Gutmann, E.D., Wood, A.W., Brekke, L.D., Arnold, J.R., 2015a. The structure for unifying multiple modeling alternatives (SUMMA), Version 1.0: Technical description.

Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J.E., Gutmann, E.D., Wood, A.W., Brekke, L.D., Arnold, J.R., Gochis, D.J., Rasmussen, R.M., 2015b. A unified approach for process-based hydrologic modeling: 1. Modeling concept. Water Resour. Res. 51, 2498–2514. https://doi.org/10.1002/2015WR017198

Clark, M.P., Nijssen, B., Lundquist, J.D., Kavetski, D., Rupp, D.E., Woods, R.A., Freer, J.E., Gutmann, E.D., Wood, A.W., Gochis, D.J., Rasmussen, R.M., Tarboton, D.G., Mahat, V., Flerchinger, G.N., Marks, D.G., 2015c. A unified approach for process-based hydrologic modeling: 2. Model implementation and case studies. Water Resour. Res. 51, 2515–2542. https://doi.org/10.1002/2015WR017200

Clark, M.P., Vogel, R.M., Lamontagne, J.R., Mizukami, N., Knoben, W.J., Tang, G., Gharari, S., Freer, J.E., Whitfield, P.H., Shook, K.R., Papalexiou, S.M., 2021. The abuse of popular performance metrics in hydrologic modeling. Water Resour. Res. 57, p.e2020WR029001. https://doi.org/https://doi.org/10.1029/2020WR029001

Coyle, K., Baker, T., 2008. Guidelines for Dublin Core Application Profiles [WWW Document]. URL https://www.dublincore.org/specifications/dublin-core/profile-guidelines/ (accessed 5.20.22).

CyberGIS Center HydroShare Development Team, 2022. CyberGIS-Jupyter for Water (CJW) Announcements [WWW Document]. URL http://www.hydroshare.org/resource/a901d83a1281404fae58cc41c1cc9889

Dublin Core Metadata Initiative. Dublin core metadata element set, version 1.1. [WWW Document], 2012. URL https://www.dublincore.org/specifications/dublin-core/dces/ (accessed 5.20.22).

Duval, E., Hodgins, W., Sutton, S., Weibel, S.L., 2002. Metadata principles and practicalities. D-Lib Mag. 8, 1–10. https://doi.org/10.1045/april2002-weibel

Ercan, M.B., Maghami, I., Bowes, B.D., Morsy, M.M., Goodall, J., 2020. Estimating Potential Climate Change Effects on the Upper Neuse Watershed Water Balance Using the SWAT Model. J. Am. Water Resour. Assoc. 56, 53–67. https://doi.org/10.1111/1752-1688.12813

Essawy, B.T., Goodall, J., Voce, D., Morsy, M.M., Sadler, J.M., Choi, Y.-D., Tarboton, D.G., Malik, T., 2020. A taxonomy for reproducible and replicable research in environmental modelling. Environ. Model. Softw. 134. https://doi.org/10.1016/j.envsoft.2020.104753

Essawy, B.T., Goodall, J., Zell, W., Voce, D., Morsy, M.M., Sadler, J., Yuan, Z., Malik, T., 2018. Integrating scientific cyberinfrastructures to improve reproducibility in computational hydrology: Example for HydroShare and GeoTrust. Environ. Model. Softw. 105, 217–229. https://doi.org/10.1016/j.envsoft.2018.03.025

Essawy, B.T., Goodall, J.L., Xu, H., Gil, Y., 2017. Evaluation of the OntoSoft Ontology for describing metadata for legacy hydrologic modeling software. Environ. Model. Softw. 92, 317–329. https://doi.org/https://doi.org/10.1016/j.envsoft.2017.01.024

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 Sp. Sci. 3, 163–175. https://doi.org/https://doi.org/10.1002/2015EA000139

Foster, I., 2011. Globus Online: Accelerating and democratizing science through cloud-based services. IEEE Internet Comput. 15, 70–73. https://doi.org/10.1109/MIC.2011.64

Gan, T., Tarboton, D.G., Dash, P., Gichamo, T.Z., Horsburgh, J.S., 2020a. Integrating hydrologic modeling web services with online data sharing to prepare, store, and execute hydrologic models. Environ. Model. Softw. 130. https://doi.org/https://doi.org/10.1016/j.envsoft.2020.104731

Gan, T., Tarboton, D.G., Horsburgh, J.S., Dash, P., Idaszak, R., Yi, H., 2020b. Collaborative sharing of multidimensional space-time data in a next generation hydrologic information system. Environ. Model. Softw. 129. https://doi.org/10.1016/j.envsoft.2020.104706

Gichamo, T.Z., Sazib, N.S., Tarboton, D.G., Dash, P., 2020. HydroDS: data services in support of physically based, distributed hydrological models. Environ. Model. Softw. 125, 104623. https://doi.org/https://doi.org/10.1016/j.envsoft.2020.104623

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J., Karlstrom, L., Lee, H., Mills, H.J., Oh, J., Pierce, S.A., 2016. Toward the Geoscience Paper of the Future: Best practices for documenting and sharing research from data to software to provenance. Earth Sp. Sci. 388–415. https://doi.org/10.1002/2015EA000136.Received

Gil, Y., Ratnakar, V., Garijo, D., 2015. OntoSoft: Capturing Scientific Software Metadata 1–4. https://doi.org/10.1145/2815833.2816955

Greenberg, J., Swauger, S., Feinstein, E.M., 2013. Metadata Capital in a Data Repository. Metadata Reuse: A Growth Indicator., in: Proc. of the International Conference on Dublin Core and Metadata Applications. pp. 140–150.

Gupta, H.V., Kling, H., Yilmaz, K.K., Martinez, G.F., 2009. Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. J. Hydrol. 377, 80–91. https://doi.org/https://doi.org/10.5194/hess-23-4323-2019.

Hancock, D.Y., Fischer, J., Lowe, J.M., Snapp-Childs, W., Pierce, M., Marru, S., Coulter, J.E., Vaughn, M., Beck, B., Merchant, N., Skidmore, E., Jacobs, G., 2021. Jetstream2: Accelerating cloud computing via Jetstream, in: Practice and Experience in Advanced Research Computing (PEARC '21). Association for Computing Machinery, New York, NY, USA, p. Article 11, 1–8. https://doi.org/https://doi.org/10.1145/3437359.3465565

Hill, L.L., Crosier, S.J., Smith, T.R., Goodchild, M., 2001. A content standard for computational models. D-Lib Mag. 7, 1082–9873.

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J., Gan, T., Yi, H., Stealey, M.J., Tarboton, D.G., 2016. HydroShare: Sharing Diverse Environmental Data Types and Models as Social Objects with Application to the Hydrology Domain. J. Am. Water Resour. Assoc. 52. https://doi.org/10.1111/1752-1688.12363

Hutton, C., Wagener, T., Freer, J., Han, D., Duffy, C., Arheimer, B., 2016. Most computational hydrology is not reproducible, so is it really science? Water Resour. Res. 52, 7548–7555. https://doi.org/10.1002/2016WR019285

IETF, 2013. JSON Schema: core definitions and terminology draft-zyp-json-schema-04 [WWW Document]. Internet-Drafts are Work. Doc. Internet Eng. Task Force. URL doc/html/draft-zyp-json-schema-04 (accessed 11.9.21).

Kurtz, W., Lapin, A., Schilling, O.S., Tang, Q., Schiller, E., Braun, T., Hunkeler, D., Vereecken, H., Sudicky, E., Kropf, P., Franssen, H.J.H., 2017. Integrating hydrological modelling, data assimilation and cloud computing for real-time management of water resources. Environ. Model. Softw. 93, 418–435. https://doi.org/https://doi.org/10.1016/j.envsoft.2017.03.011

Kurtzer, G.M., Sochat, V., Bauer, M.W., 2017. Singularity: Scientific containers for mobility of compute. PLoS One 12, e0177459. https://doi.org/10.1371/journal.pone.0177459

Leipzig, J., Nüst, D., Hoyt, C.T., Ram, K., Greenberg, J., 2021. The role of metadata in reproducible computational research. Patterns 2, 100322. https://doi.org/10.1016/j.patter.2021.100322

Li, Z., Michels, A., Lu, F., Padmanabhan, A., Wang, S., 2022. CyberGIS-Jupyter for Water [WWW Document]. HydroShare. URL http://www.hydroshare.org/resource/4cfd280e8eb747169b293aec2862d4f5

Lyu, F., Yin, D., Padmanabhan, A., Choi, Y.-D., Goodall, J.L., Castronova., A.M., Tarboton, D.G., Wang, S., 2019. Reproducible hydrological modeling with CyberGIS-Jupyter: a case study on SUMMA, in: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning). pp. 1–6. https://doi.org/https://doi.org/10.1145/3332186.3333052

Maghami, I., 2022. The revised resources created by Choi (2020) as: “Toward Open and Reproducible Environmental Modeling by Integrating Online Data Repositories, Computational Environments, and Model Application Programming Interfaces” [WWW Document]. HydroShare. URL https://www.hydroshare.org/resource/8c499dadaa274af488775596212ebf73/ (accessed 10.7.22).

Maghami, I., 2021. Data for Estimating Potential Climate Change Effects on the Upper Neuse Watershed Water Balance Using the SWAT Model [WWW Document]. URL https://www.hydroshare.org/resource/1710a6b553da4f70877c2abc69f3a42b/

Mathevet, T., Michel, C., Andréassian, V., Perrin, C., 2006. A bounded version of the Nash-Sutcliffe criterion for better model assessment on large sets of basins, in: Large Sample

Basin Experiments for Hydrological Model Parameterization: Results of the Model Parameter Experiment–MOPEX. IAHS PUBLICATION, Wallingford, UK, pp. 211–219.

Melsen, L.A., Torfs, P.J.J.F., Uijlenhoet, R., Teuling, A.J., 2017. Comment on “Most computational hydrology is not reproducible, so is it really science?” by Christopher Hutton et al.: Let hydrologists learn the latest computer science by working with Research Software Engineers (RSEs) and not reinvent the waterwheel our. Water Resour. Res. https://doi.org/https://doi.org/10.1002/2016WR020208

Merkel, D., 2014. Docker: lightweight Linux containers for consistent development and deployment. Linux J.

Mizukami, N., Wood, A., 2021. NLDAS Forcing NetCDF using CAMELS datasets from 1980 to 2018 [WWW Document]. HydroShare. URL http://www.hydroshare.org/resource/a28685d2dd584fe5885fc368cb76ff2a

Morsy, M.M., Goodall, J., Bandaragoda, C., Castronova, A.M., Greenberg, J., 2014. Metadata for describing water models. Proc. - 7th Int. Congr. Environ. Model. Softw. Bold Visions Environ. Model. iEMSs 2014 1, 53–59.

Morsy, M.M., Goodall, J., Castronova, A.M., Dash, P., Merwade, V., Sadler, J.M., Rajib, M.A., Horsburgh, J.S., Tarboton, D.G., 2017. Design of a metadata framework for environmental models with an example hydrologic application in HydroShare. Environ. Model. Softw. 93, 13–28. https://doi.org/10.1016/j.envsoft.2017.02.028

Newman, A.J., Clark, M.P., Craig, J., Nijssen, B., Wood, A.W., Gutmann, E.D., Mizukami, N., Brekke, L., Arnold, J.R., 2015a. Gridded ensemble precipitation and temperature estimates for the contiguous United States. J. Hydrometeorol. 16, 2481–2500. https://doi.org/https://doi.org/10.1175/JHM-D-15-0026.1

Newman, A.J., Clark, M.P., Sampson, K., Wood, A., Hay, L.E., Bock, A., Viger, R.J., Blodgett, D., Brekke, L., Arnold, J.R., Hopson, T., Duan, Q., 2015b. Development of a large-sample watershed-scale hydrometeorological data set for the contiguous USA: Data set characteristics and assessment of regional variability in hydrologic model performance. Hydrol. Earth Syst. Sci. 19, 209–223. https://doi.org/https://doi.org/10.5194/hess-19-209-2015, 2015

Nüst, D., Granell, C., Hofer, B., Konkol, M., Ostermann, F.O., Sileryte, R., Cerutti, V., 2018. Reproducible research and GIScience: An evaluation using AGILE conference papers. PeerJ 2018, 1–23. https://doi.org/10.7717/peerj.5072

Obels, P., Lakens, D., Coles, N.A., Gottfried, J., Green, S.A., 2020. Analysis of Open Data and Computational Reproducibility in Registered Reports in Psychology. Adv. Methods Pract. Psychol. Sci. 3, 229–237. https://doi.org/10.1177/2515245920918872

Ochiai, K., Nagamori, M., Sugimoto, S., 2014. A metadata schema design model and support system based on an agile development model, in: IConference 2014 Proceedings.

Overeem, I., Berlin, M.M., Syvitski, J.P.M., 2013. Strategies for integrated modeling: The community surface dynamics modeling system example. Environ. Model. Softw. 39, 314–321. https://doi.org/https://doi.org/10.1016/j.envsoft.2012.01.012

Peckham, S.D., 2014. The CSDMS Standard Names: Cross-Domain Naming Conventions for Describing Process Models, Data Sets and Their Associated Variables, in: Ames, D.P., Quinn, N.W.T., Rizzoli, A.E. (Eds.), 7th Intl. Congress on Env. Modeling and Software. International Environmental Modelling and Software Society (IEMSs), San Diego, California, USA.

Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., Vrgoč, D., 2016. Foundations of JSON schema, in: Proceedings of the 25th International Conference on World Wide Web. pp. 263–273. https://doi.org/10.1145/2872427.2883029

Piccolo, S.R., Frampton, M.B., 2016. Tools and techniques for computational reproducibility. Gigascience 5, s13742-016. https://doi.org/10.1101/045104.Gronenschild

Qin, J., Dobreski, B., Brown, D., 2016. Metadata and Reproducibility: A Case Study of Gravitational Wave Research Data Management. Int. J. Digit. Curation 11, 218–231. https://doi.org/10.2218/ijdc.v11i1.399

Rajib, M.A., Merwade, V., Kim, I.L., Zhao, L., Song, C., Zhe, S., 2016. SWATShare – A web platform for collaborative research and education through online sharing, simulation and visualization of SWAT models. Environ. Model. Softw. 75, 498–512. https://doi.org/https://doi.org/10.1016/j.envsoft.2015.10.032

Riddick, A.T., Heaven, R., Royse, K.R., Singh, A., Hughes, A.G., 2020. A model metadata schema for environmental hazard models and its implementation in the PURE portal. Environ. Model. Softw. 124, 1–15. https://doi.org/10.1016/j.envsoft.2019.104597

Stagge, J.H., Rosenberg, D.E., Abdallah, A.M., Akbar, H., Attallah, N.A., James, R., 2019. Assessing data availability and research reproducibility in hydrology and water resources. Sci. Data 6, 1–12. https://doi.org/10.1038/sdata.2019.30

Stewart, C.A., Cockerill, T.M., Foster, I., Hancock, D., Merchant, N., Skidmore, E., Stanzione, D., Taylor, J., Tuecke, S., Turner, G., Vaughn, M., Gaffney, N.I., 2015. Jetstream: a self-provisioned, scalable science and engineering cloud environment, in: Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure. St. Louis, Missouri, p. ACM: 2792774. p. 1-8. https://doi.org/https://dx.doi.org/10.1145/2792745.2792774

Stodden, V., Bailey, D., Borwein, J., 2013. Setting the Default to Reproducible. Comput. Sci. Res. 46, 4–6.

Stodden, V., Miguez, S., Seiler, J., 2015. Researchcompendia. org: Cyberinfrastructure for reproducibility and collaboration in computational science. Comput. Sci. Eng. 17, 12–19. https://doi.org/10.1109/MCSE.2015.18

Tarboton, D., Calloway, C., 2021. THREDDS DAP2 [WWW Document]. URL http://www.hydroshare.org/resource/70070fa1b382496e85ca44894683b15d

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Ames, D.P., Goodall, J.L., Band, L.E., Merwade, V., Couch, A., Hooper, R.P., Maidment, D.R., Dash, P.K., 2014a. HydroShare: Advancing collaboration through hydrologic data and model sharing, in: Proceedings of the 7th International Congress on Environmental Modelling and Software. Int. Environ. Modell.

and Software Soc, San Diego, Calif., pp. 978–988.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J., Band, L.E., Merwade, V., 2014b. A Resource Centric Approach For Advancing Collaboration Through Hydrologic Data And Model Sharing, in: International Conference on Hydroinformatics. CUNY Academic Works.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Merwade, V., Couch, A.L., Arrigo, J., Hooper, R., Valentine, D., Maidment, D., 2014c. HydroShare: Advancing Hydrology through Collaborative Data and Model Sharing, in: International Environmental Modelling and Software Society (IEMSs) 7th International Congress on Environmental Modelling and Software.

Tayfur, G., 2017. Modern Optimization Methods in Water Resources Planning, Engineering and Management. Water Resour. Manag. 31, 3205–3233. https://doi.org/10.1007/s11269-017-1694-6

The International Organization for Standardization; ISO 23081.1—s3 Terms and Definitions [WWW Document], 2017. URL https://www.iso.org/obp/ui/#iso:std:iso:23081:-1:ed-2:v1:en (accessed 5.20.22).

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G.D., Roskies, R., Scott, J.R., Wilkins-Diehr, N., 2014. XSEDE: Accelerating Scientific Discovery. Comput. Sci. Eng. 16, 62–74. https://doi.org/https://dx.doi.org/10.1109/MCSE.2014.80

Van Beusekom, A.E., Hay, L.E., Bennett, A.R., Choi, Y.-D., Clark, M.P., Goodall, J., Li, Z., Maghami, I., Nijssen, B., Wood, A.W., 2022. Hydrologic Model Sensitivity to Temporal Aggregation of Meteorological Forcing Data: A Case Study for the Contiguous United States. J. Hydrometeorol. 23, 167–183. https://doi.org/10.1175/JHM-D-21-0111.1

Where to start — advice on creating a metadata schema [WWW Document], 2014. . ISO/TC 46/SC11N800R1. URL https://committee.iso.org/files/live/sites/tc46sc11/files/documents/N800R1 Where to start-advice on creating a metadata schema.pdf (accessed 12.7.22).

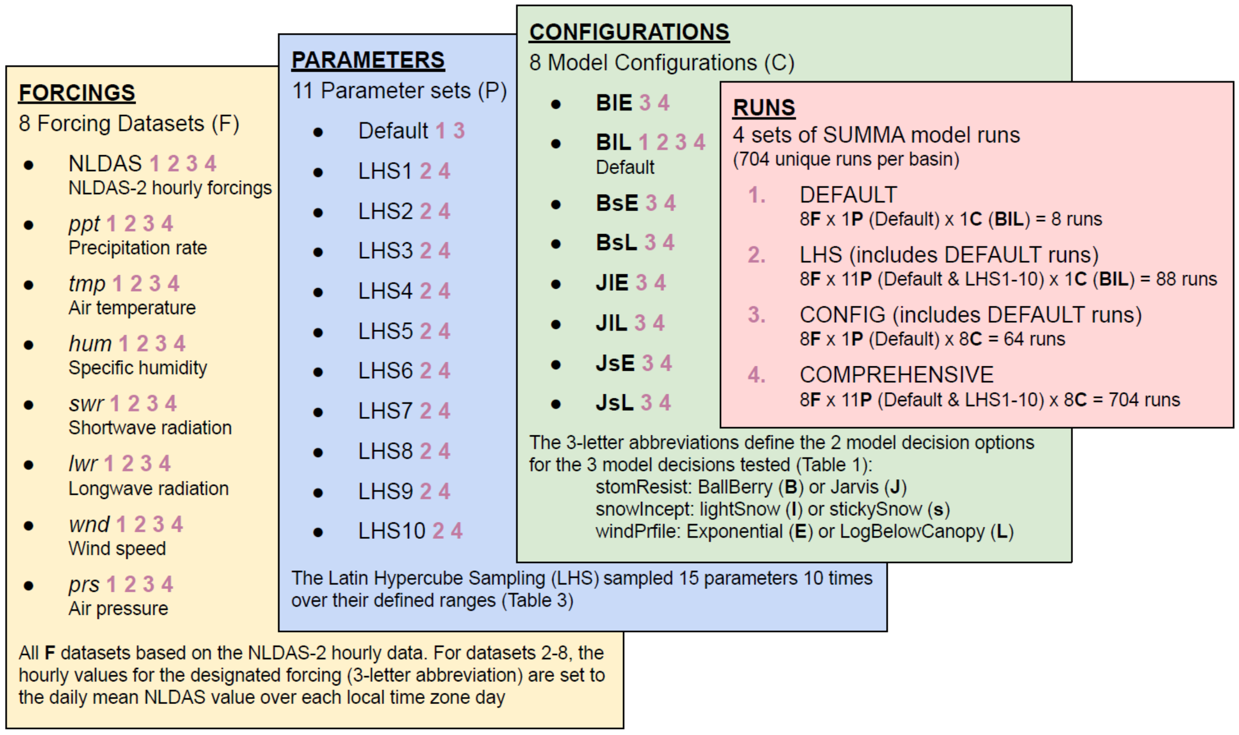
Wosniok, C., Lehfeldt, R., 2013. A metadata-driven management system for numerical modeling, in: OCEANS 2013 MTS/IEEE - San Diego: An Ocean in Common. MTS.

Yang, C., Raskin, R., Goodchild, M., Gahegan, M., 2010. Geospatial cyberinfrastructure: past, present and future. Computers, Environment and Urban Systems. Comput. Environ. Urban Syst. 34, 264–277. https://doi.org/https://doi.org/10.1016/j.compenvurbsys.2010.04.001

Appendix

Appendix-1

This section provides supplemental material to support our methods and results for Chapter 4. The figures and tables are referred to in the main text.



FORCINGS
8 Forcing Datasets (F)

- NLDAS 1 2 3 4
  NLDAS-2 hourly forcings
- *ppt* 1 2 3 4
  Precipitation rate
- *tmp* 1 2 3 4
  Air temperature
- *hum* 1 2 3 4
  Specific humidity
- *swr* 1 2 3 4
  Shortwave radiation
- *lwr* 1 2 3 4
  Longwave radiation
- *wnd* 1 2 3 4
  Wind speed
- *prs* 1 2 3 4
  Air pressure

All **F** datasets based on the NLDAS-2 hourly data. For datasets 2-8, the hourly values for the designated forcing (3-letter abbreviation) are set to the daily mean NLDAS value over each local time zone day

PARAMETERS
11 Parameter sets (P)

- Default 1 3
- LHS1 2 4
- LHS2 2 4
- LHS3 2 4
- LHS4 2 4
- LHS5 2 4
- LHS6 2 4
- LHS7 2 4
- LHS8 2 4
- LHS9 2 4
- LHS10 2 4

The Latin Hypercube Sampling (LHS) sampled 15 parameters 10 times over their defined ranges (Table 3)

CONFIGURATIONS
8 Model Configurations (C)

- **BIE** 3 4
- **BIL** 1 2 3 4
  Default
- **BsE** 3 4
- **BsL** 3 4
- **JIE** 3 4
- **JIL** 3 4
- **JsE** 3 4
- **JsL** 3 4

The 3-letter abbreviations define the 2 model decision options for the 3 model decisions tested (Table 1):
stomResist: BallBerry (**B**) or Jarvis (**J**)
snowIncept: lightSnow (**l**) or stickySnow (**s**)
windPrfile: Exponential (**E**) or LogBelowCanopy (**L**)

RUNS
4 sets of SUMMA model runs
(704 unique runs per basin)

1. DEFAULT
   8**F** x 1**P** (Default) x 1**C** (**BIL**) = 8 runs
2. LHS (includes DEFAULT runs)
   8**F** x 11**P** (Default & LHS1-10) x 1**C** (**BIL**) = 88 runs
3. CONFIG (includes DEFAULT runs)
   8**F** x 1**P** (Default) x 8**C** = 64 runs
4. COMPREHENSIVE
   8**F** x 11**P** (Default & LHS1-10) x 8**C** = 704 runs

Figure A1. An overview of the forcing datasets (FORCINGS; yellow box), parameter sets (PARAMETERS; blue box), and model configurations (CONFIGURATIONS; green box) used in the 704 SUMMA model runs (RUNS; pink box) performed for each of the 671 CAMELS basins. Note the pink numbers that follow each forcing, parameter, and configuration refers to the SUMMA model run set as numbered in the pink RUNS box (e.g., the Default parameter set in the PARAMETERS box is used with SUMMA model runs 1 and 3 in the RUNS box) (source: Van Beusekom et al., 2022)

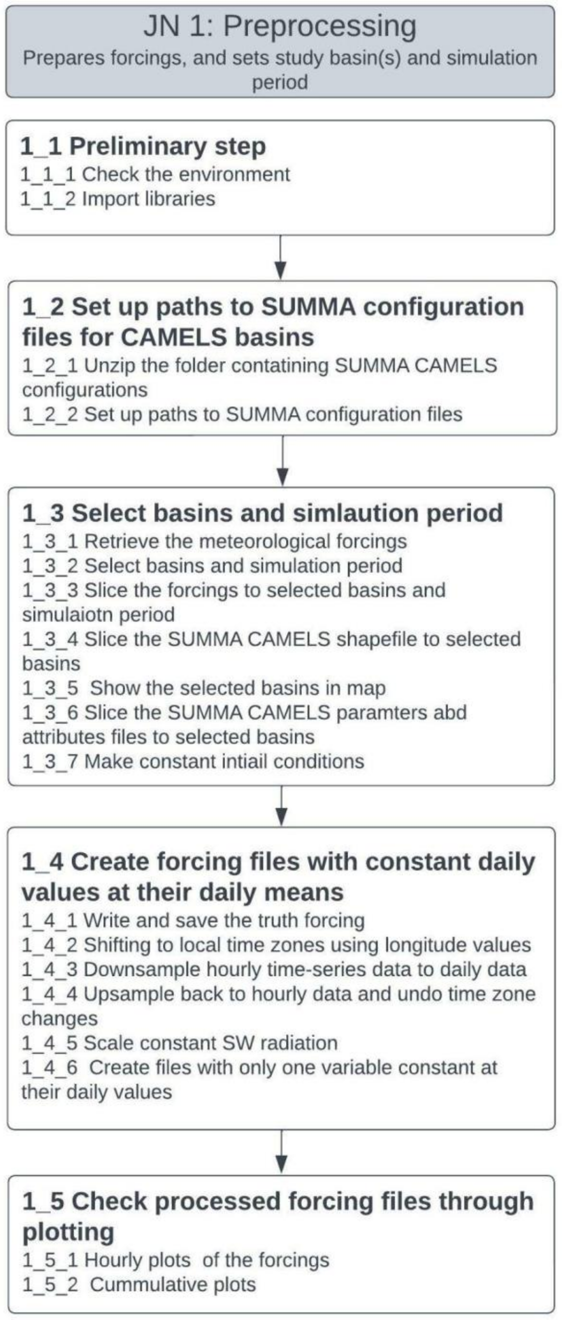

JN 1: Preprocessing
Prepares forcings, and sets study basin(s) and simulation period
1_1 Preliminary step
1_1_1 Check the environment
1_1_2 Import libraries
1_2 Set up paths to SUMMA configuration files for CAMELS basins
1_2_1 Unzip the folder contatining SUMMA CAMELS configurations
1_2_2 Set up paths to SUMMA configuration files
1_3 Select basins and simlaution period
1_3_1 Retrieve the meteorological forcings
1_3_2 Select basins and simulation period
1_3_3 Slice the forcings to selected basins and simulaiotn period
1_3_4 Slice the SUMMA CAMELS shapefile to selected basins
1_3_5 Show the selected basins in map
1_3_6 Slice the SUMMA CAMELS paramters abd attributes files to selected basins
1_3_7 Make constant intiail conditions
1_4 Create forcing files with constant daily values at their daily means
1_4_1 Write and save the truth forcing
1_4_2 Shifting to local time zones using longitude values
1_4_3 Downsample hourly time-series data to daily data
1_4_4 Upsample back to hourly data and undo time zone changes
1_4_5 Scale constant SW radiation
1_4_6 Create files with only one variable constant at their daily values
1_5 Check processed forcing files through plotting
1_5_1 Hourly plots of the forcings
1_5_2 Cummulative plots

Figure A2. The preprocessing notebook (JN1) diagram

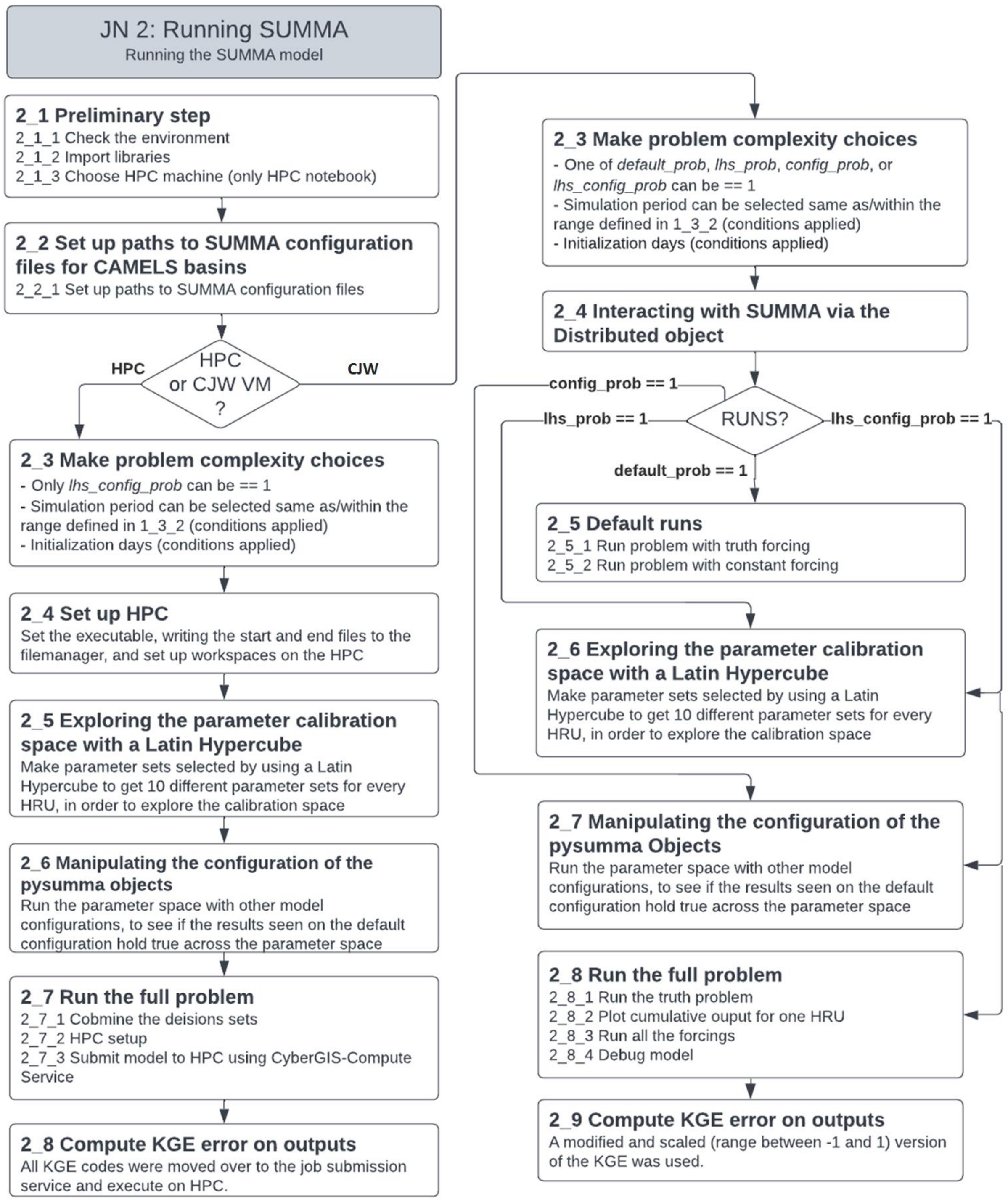

JN 2: Running SUMMA
Running the SUMMA model
2_1 Preliminary step
2_1_1 Check the environment
2_1_2 Import libraries
2_1_3 Choose HPC machine (only HPC notebook)
2_2 Set up paths to SUMMA configuration files for CAMELS basins
2_2_1 Set up paths to SUMMA configuration files
HPC or CJW VM ?
HPC
CJW
2_3 Make problem complexity choices
- Only lhs_config_prob can be == 1
- Simulation period can be selected same as/within the range defined in 1_3_2 (conditions applied)
- Initialization days (conditions applied)
2_4 Set up HPC
Set the executable, writing the start and end files to the filemanager, and set up workspaces on the HPC
2_5 Exploring the parameter calibration space with a Latin Hypercube
Make parameter sets selected by using a Latin Hypercube to get 10 different parameter sets for every HRU, in order to explore the calibration space
2_6 Manipulating the configuration of the pysumma objects
Run the parameter space with other model configurations, to see if the results seen on the default configuration hold true across the parameter space
2_7 Run the full problem
2_7_1 Cobmine the deisions sets
2_7_2 HPC setup
2_7_3 Submit model to HPC using CyberGIS-Compute Service
2_8 Compute KGE error on outputs
All KGE codes were moved over to the job submission service and execute on HPC.
2_3 Make problem complexity choices
- One of default_prob, lhs_prob, config_prob, or lhs_config_prob can be == 1
- Simulation period can be selected same as/within the range defined in 1_3_2 (conditions applied)
- Initialization days (conditions applied)
2_4 Interacting with SUMMA via the Distributed object
RUNS?
config_prob == 1
lhs_prob == 1
lhs_config_prob == 1
default_prob == 1
2_5 Default runs
2_5_1 Run problem with truth forcing
2_5_2 Run problem with constant forcing
2_6 Exploring the parameter calibration space with a Latin Hypercube
Make parameter sets selected by using a Latin Hypercube to get 10 different parameter sets for every HRU, in order to explore the calibration space
2_7 Manipulating the configuration of the pysumma Objects
Run the parameter space with other model configurations, to see if the results seen on the default configuration hold true across the parameter space
2_8 Run the full problem
2_8_1 Run the truth problem
2_8_2 Plot cumulative ouput for one HRU
2_8_3 Run all the forcings
2_8_4 Debug model
2_9 Compute KGE error on outputs
A modified and scaled (range between -1 and 1) version of the KGE was used.

Figure A3. Running SUMMA notebook (JN2) diagram

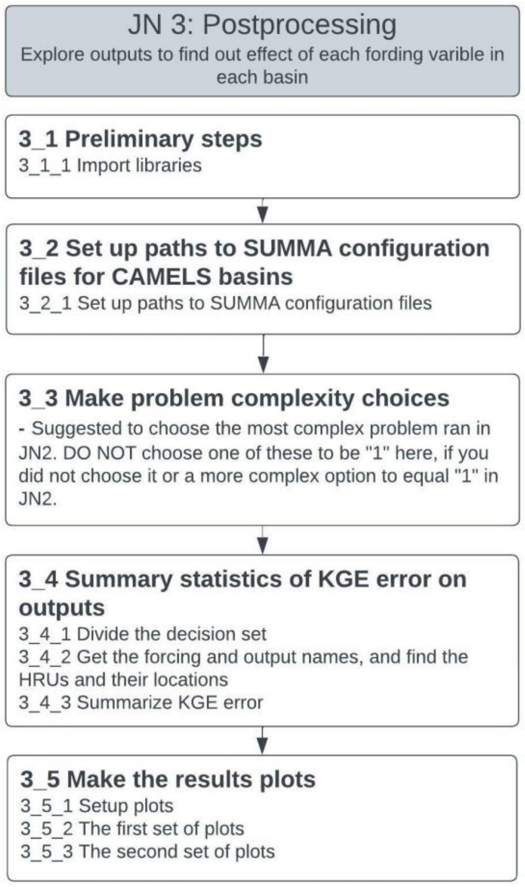

JN 3: Postprocessing
Explore outputs to find out effect of each fording varible in each basin
3_1 Preliminary steps
3_1_1 Import libraries
3_2 Set up paths to SUMMA configuration files for CAMELS basins
3_2_1 Set up paths to SUMMA configuration files
3_3 Make problem complexity choices
- Suggested to choose the most complex problem ran in JN2. DO NOT choose one of these to be "1" here, if you did not choose it or a more complex option to equal "1" in JN2.
3_4 Summary statistics of KGE error on outputs
3_4_1 Divide the decision set
3_4_2 Get the forcing and output names, and find the HRUs and their locations
3_4_3 Summarize KGE error
3_5 Make the results plots
3_5_1 Setup plots
3_5_2 The first set of plots
3_5_3 The second set of plots

Figure A4. Post-processing notebook (JN3) diagram

Appendix

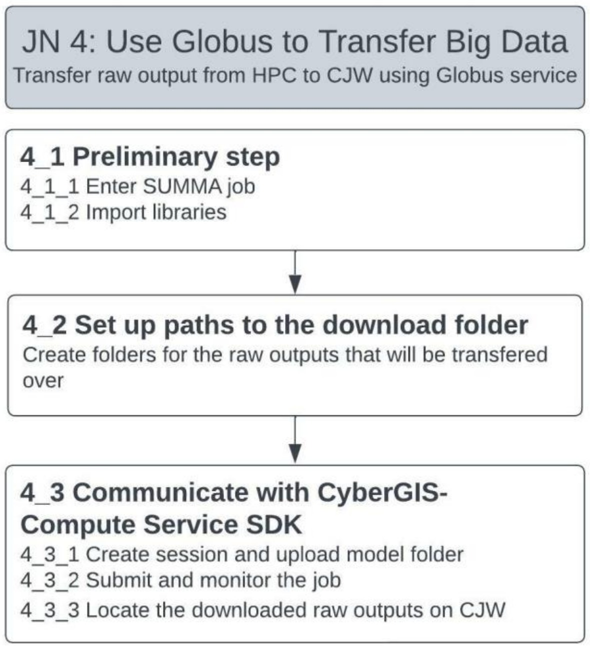

JN 4: Use Globus to Transfer Big Data
Transfer raw output from HPC to CJW using Globus service
4_1 Preliminary step
4_1_1 Enter SUMMA job
4_1_2 Import libraries
4_2 Set up paths to the download folder
Create folders for the raw outputs that will be transfered over
4_3 Communicate with CyberGIS-Compute Service SDK
4_3_1 Create session and upload model folder
4_3_2 Submit and monitor the job
4_3_3 Locate the downloaded raw outputs on CJW

Figure A5. HPC Data transfer notebook (JN4) diagram

Table A1. SUMMA output variables chosen for analysis (source: Van Beusekom et al., 2022)

#	Variable Type	SUMMA Variable Name	Description (units)
1	liquid water fluxes for the soil domain	SurfaceRunoff	surface runoff (m s-1)
2		AquiferBaseflow	baseflow from the aquifer (m s-1)
3		Infiltration	infiltration of water into the soil profile (m s-1)
4		RainPlusMelt	rain plus melt (m s-1)
5		SoilDrainage	drainage from the bottom of the soil profile (m s-1)
6	turbulent heat transfer	LatHeatTotal	latent heat from the canopy air space to the atmosphere (W m-2)
7		SenHeatTotal	sensible heat from the canopy air space to the atmosphere (W m-2)
8		SnowSublimation	snow sublimation/frost (below canopy or non-vegetated) (kg m-2 s-1)
9	snow	SWE	snow water equivalent (kg m-2)
10	vegetation	CanopyWat	mass of total water on the vegetation canopy (kg m-2)
11	derived	NetRadiation	net radiation (W m-2)
12		TotalET	total evapotranspiration (kg m-2 s-1)
13		TotalRunoff	total runoff (m s-1)
14		TotalSoilWat	total mass of water in the soil (kg m-2)

Table A2. Parameters chosen for Latin Hypercube Sampling (source: Van Beusekom et al., 2022)

Parameter Name	Minimum	Maximum	Default	Constraints
k_macropore	1.0d-7	0.1	0.0001	
k_soil	1.0d-7	1.0d-5	variable	
theta_sat	0.3	0.6	variable	> critSoilTranspire; > fieldCapacity; > theta_res
aquiferBaseflowExp	1	10	2.0	
aquiferBaseflowRate	0	0.1	0.1	
qSurfScale	1	100	50	
summerLAI	0.01	10	3	
frozenPrecipMultip	0.5	1.5	1	
heightCanopyTop	0.05	100	variable	> heightCanopyBottom
heightCanopyBottom	0	5	variable	
routingGammaShape	2	3	2.5	
routingGammaScale	1	100000	20000	
albedoRefresh	1	10	1.0	
tempCritRain	272.16	274.16	273.16	
windReductionParam	0	1	0.28	

The eight plots generated by Notebook 3 are described as follows:

1. Location of the selected CAMELS basin.
2. KGE values for each CNST forcing dataset (datasets 2-8; Table A2) by output variable using the DEFAULT model runs. This is a subset of Figure 9A from Van Beusekom et al. (2022).
3. Boxplots depicting the range in the KGE values for each set of model runs (DEFAULT, LHS, CONFIG, and COMPREHENSIVE; Table A1) by output variable. Note, boxplots only appear for the model runs selected in Notebook 2. This is a subset of Figure 9B from Van Beusekom et al. (2022).
4. Boxplots depicting the range in the KGE values for each set of model runs (DEFAULT, LHS, CONFIG, COMPREHENSIVE; Table A1) by CNST forcing dataset (datasets 2-8; Table A2). Note, boxplots only appear for the model runs executed in Notebook 2. This is a subset of Figure 9C from Van Beusekom et al. (2022).
5. Ranks 1 - 7 stacked barplots depicting the relative basin KGE rank counts by CNST forcing dataset (datasets 2-8; Table A2) for the 14 SUMMA output variables. Note, bars on this

plot will only appear if the COMPREHENSIVE basin runs are executed in Notebook 2. This is a subset of Figure 8 from Van Beusekom et al. (2022).

6. Ranks 1 - 7 stacked barplots depicting the relative basin KGE rank counts by CNST forcing dataset (datasets 2-8; Table A2) for the eight SUMMA configurations. Note, the complete figure will only appear if the COMPREHENSIVE basin runs are executed in Notebook 2. A stacked bar for the default configuration (BlL) will be plotted if the LHS basin runs are executed in Notebook 2. This is a subset of Figure 8 from Van Beusekom et al. (2022).
7. Boxplots for each output variable depicting the range in the seven-summed KGE values (from CNST forcing datasets 2-8) for the eight SUMMA configurations, or for the default configuration if only the default configuration was run (DEFAULT or LHS basin runs in Notebook 2. This is a subset of Figure 6 from Van Beusekom et al. (2022).
8. Boxplots depicting the range in the summed SUMMA hourly output variables over the period of record produced using the benchmark (NLDAS) forcing dataset for the eight SUMMA configuration, or for the default configuration if only the default configuration was run (DEFAULT or LHS basin runs in Notebook 2). Note, a point will appear instead of a boxplot if only the default parameter set was run (DEFAULT or CONFIG basin runs in Notebook 2). This analysis is not in Van Beusekom et al. (2022); it is included in the interactive tool to supply users with potential SUMMA output variable ranges for their selected basin.