# Advancing Cyberinfrastructure for Reproducible Hydrologic 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**

**Doctor of Philosophy**
**in**
**Civil and Environmental Engineering**

**By**

**Bakinam Tarik Essawy**

**August 2017**

# APPROVAL SHEET

**The Dissertation**
**is submitted in Partial Fulfillment of the Requirements**
**for the Degree of**
**Doctor of Philosophy**

Bakinam Tarik Essawy

Author

**The dissertation has been read and approved by the examining committee:**

Jonathan Goodall, Ph.D., P.E.

Advisor

Teresa Culver, Ph.D.

John Porter, Ph.D.

Marty Humphrey, Ph.D.

Tanu Malik, Ph.D.

**Accepted for the School of Engineering and Applied Science:**

Craig H. Benson, Dean, School of Engineering and Applied Science

**August 2017**

# Abstract

Scientists have created a significant and growing collection of software tools for data manipulation, analysis, and simulation. This software includes not only computational models, but also a large collection of data pre- and post-processing tools used to support computational modeling and data analysis. This large and diverse set of scientific modeling software presents challenges to the hydrologic science community including (1) the difficulty of making scientific workflows reproducible in support of an end-to-end hydrologic modeling use case, especially when large dataset transfers are required for data processing pipelines, (2) providing adequate and accessible component-level metadata for legacy hydrologic software, and (3) ensuring scientific reproducibility when using legacy hydrologic processing software, which often has complex software dependencies. This research uses the Variable Infiltration Capacity (VIC) and MODFLOW hydrologic models as use cases in examining these challenges.

This research addresses these challenges by conducting three studies. The first study explores approaches for leveraging data grid technology in hydrologic modeling to support reproducible workflows using large datasets. Its primary contribution is a general methodology for analyzing large, distributed data collections. This is accomplished by moving processing resources to the large datasets in contrast to the typical approach of moving the datasets to the processing resources. Data grid technology is used to automate data transfers and staging, in combination with automated formal publication of generated data assets.

The second study advances prior efforts for formalizing model metadata in hydrology by evaluating the OntoSoft Ontology as a means for formally structuring model metadata for lower level scientific software components. The metadata evaluated describes a data pre-processing

workflow for the VIC hydrologic model. This workflow consists of multiple software components written by different scientists over time. The analysis begins by exploring what metadata hydrologists have already captured in unstructured forms. It then shows how this metadata could be organized into structured, machine-readable metadata using the OntoSoft Ontology.

Finally, the third study explores the creation of containers using Docker to more easily execute hydrologic modeling software in a computing environment. By containerizing a model with all of its dependencies, the model is self-contained and portable. This work contributes a methodology for using HydroShare and Geotrust, two new cyberinfrastructure tools under active development, to improve reproducibility in computational hydrology. HydroShare is a web-based system for sharing hydrologic data and model resources. GeoTrust allows scientists to document their computational workflows as containers called sciunits, a type of Docker container. sciunits include required software dependencies making execution of computational workflows more consistent across computing environments. HydroShare and GeoTrust can be used together to create open, reusable data analysis and model execution services. The services are created using GeoTrust and can be integrated with HydroShare as Web apps that operate on HydroShare resources. The MODLFOW groundwater model is used as an example to show the functionality provided by this cyberinfrastructure for creating open and reusable data analysis and model execution services.

# Table of Contents

# List of Figures

# List of Tables

## Acknowledgements

In the name of Allah, the Most Gracious and the Most Merciful. I am Grateful to the Mighty Allah for giving me the strength to complete this thesis.

I want to express sincere gratitude to **Dr. Jonathan Goodall** for all the support and encouragement throughout my graduate studies. No words can express my deep gratitude. Thank you for giving me the pleasure to be one of your graduate students.

Thanks to the service of my dissertation committee, Drs. Teresa Culver, John Porter, Marty Humphrey, and Tanu Malik. I would like to express my deep thanks to my friends **Wes Zell**, **Jeffery Saddler**, **Gina O' Neil, Ben Bowes, Yawen Shen** and **Alex Chen**. You all meant a lot to me and my family. Thank you for all your advice.

I would like to thank my husband **Dr. Mohamed Morsy** for his understanding, support and love he always provided me and still does. May Allah bless you. This work would never be done without Allah's well then you being next to me. Love you.

I would like to thank my father in law **Prof. Dr. Morsy Anwar**, and my mother in law **Reda El Nahta** for their love, and encouragements.

This work is dedicated to my parent's my father, **Prof. Dr. Tarik Essawy**, and my mother, **Dr. Nemat Eid**. Mommy and Pappy, I am always proud of you. I hope I was able to make you proud of me. I really owe you a lot. Thank you for your continuous encouragement and the love you are giving me throughout my life.

In the end, I hope that the lights of my life, **Zeina** and **Kenzy,** will be proud of their mom. Thank you, girls, so much for being with me. I am so sorry if I was sometimes too busy for you. All I want you to know is that I am all yours.

**" قَالُواْ سُبْحَانَكَ لاَ عِلْمَ لَنَا إِلاَّ مَا عَلَّمْتَنَا إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ "**

**(٣٢) سورة البقرة**

**"Qaloo subhanaka laa ailma lana illa ma aallamtana innaka anta al aaleemu alhakeem"**

In the name of God, Most Gracious, Most Merciful

They said, "Exalted are You; we have no knowledge except what You have taught us. Indeed, it is You who is the Knowing, the Wise."

Surah Al-Baqarah (32)

# Chapter 1: Introduction

Hydrologists use many different computational models, with each model tailored to address specific aspects of the hydrologic cycle. Hydrologic modeling has a long history, and many computational models have decades of development effort and many model versions behind them (Singh et al., 2002). Modelers often use additional software to prepare inputs for a specific model (pre-processing software) and to analyze outputs generated by the model (post-processing software). Taken together, there is a large and growing ecosystem of software used to simulate hydrologic systems.

Modelers often spend weeks or months building, calibrating and validating their models. This process is complicated and requires a diversity of data from many different data providers. Steps in this process are rarely automated and methods for completing these steps often involve tacit knowledge that is difficult to automate. As a result, most scientific papers describing results from a modeling study lie under the "not reproducible" category along a spectrum of reproducibility standards (Peng, 2011) Figure 1.1. A "not reproducible" publication represents a publication written by a scientist that created an experiment or study and only published the results for this experiment or study rather than the linked code and data needed to replicate the study.

The overarching objective of this research is to move hydrologic modeling toward the gold standard of "full replication" in the reproducibility spectrum. In hydrologic modeling, due to the variety and size of data and models used, the lack of metadata standards, and the extensive data pre- and post- processing steps associated with the modeling workflow, reaching full replication is nontrivial (Hutton et al., 2016). This research addresses these challenges by focusing on the

Variable Infiltration Capacity (VIC) and MODFLOW hydrologic models, popular and widely used hydrologic models with more than two decades of development effort, as a case studies.

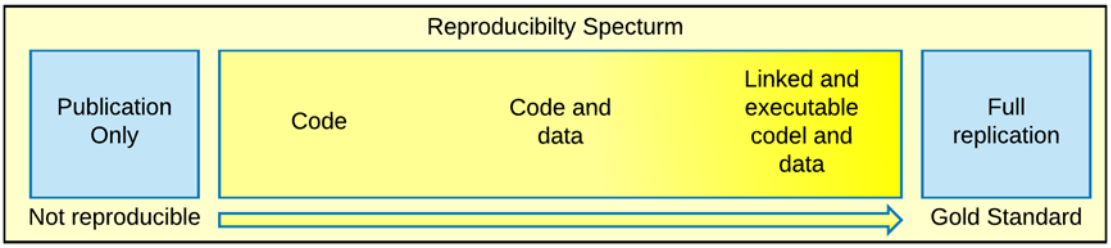



**Figure 1.1** Spectrum of reproducibility standards (Peng, 2011).

## 1.1 Background

### *1.1.1 Modeling Data Life Cycle*

Figure 1.2 shows the modeling data life cycle. This cycle starts with collecting the data, then proceeds to executing the code used to process this raw data. The processed data is then used to produce the computational results through the analytic code. These computational results are then summarized in tables, numerical results, and even visualizations that are presented in a journal article. Often the reader has access to only the final result, the published article. Without the materials and information used to produce the results published in the article, the reader is unable to verify the published results and conduct an alternate analysis of the same data (Peng and Eckel, 2009). This end-to-end workflow from collected data to the article should be more automated to better enable reproducibility, transparency, and reuse of modeling studies.

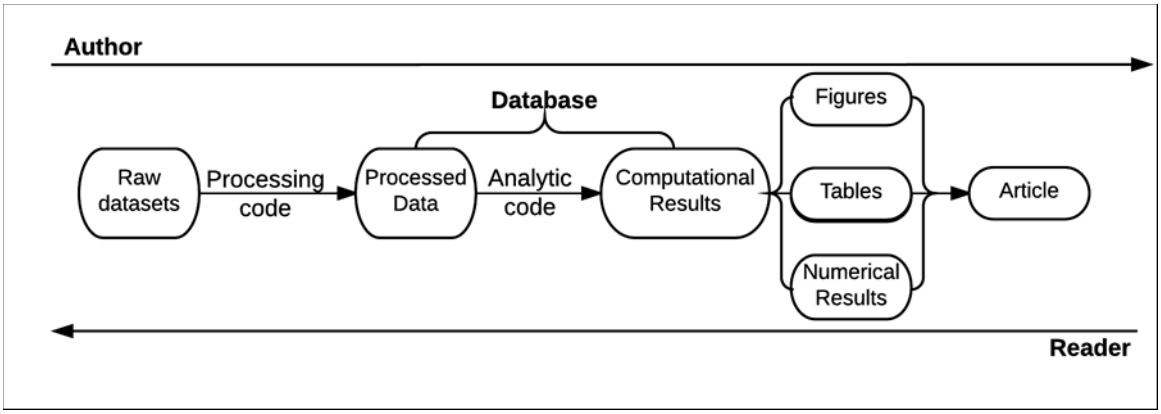



**Figure 1.2** Modeling data cycle pipeline for reproducible research (Peng & Eckel, 2009).

### *1.1.2 An Example of the Hydrologic Modeling Workflow*

The Variable Infiltration Capacity (VIC) provides an example of the workflow typical in hydrologic modeling. VIC model was developed at the University of Washington and Princeton University beginning in early 1990s. VIC is a macro scale hydrologic model that applies water and energy balances to simulate terrestrial hydrology at a regional spatial scale (Liang et al., 1996a). Figure 1.3 shows a schematic of the VIC model. The VIC model is typically run at a spatial resolution of 1/8$^{th}$ degree grid cells. The model represents the land surface as three layers of soil, and is able to simulate the land surface portion of the hydrologic cycle by solving the full water and surface energy balance equations (Liang et al., 1996a, 1996c).

Like many hydrologic models, the VIC model requires significant effort to prepare its input data. Figure 1.3 shows the data processing workflow used to generate the meteorological and land surface input datasets for a VIC model simulation. This workflow consists of a sequence of 15 data processing steps, each step requiring input datasets from different sources, and many of the datasets having unique and non-standardized data models and formats (Billah et al., 2016a). These scripts are written in different programming languages including Fortran 77, C, and C++. Shell

scripts are used throughout the workflow to execute these steps and perform other commands required to complete the data processing tasks.

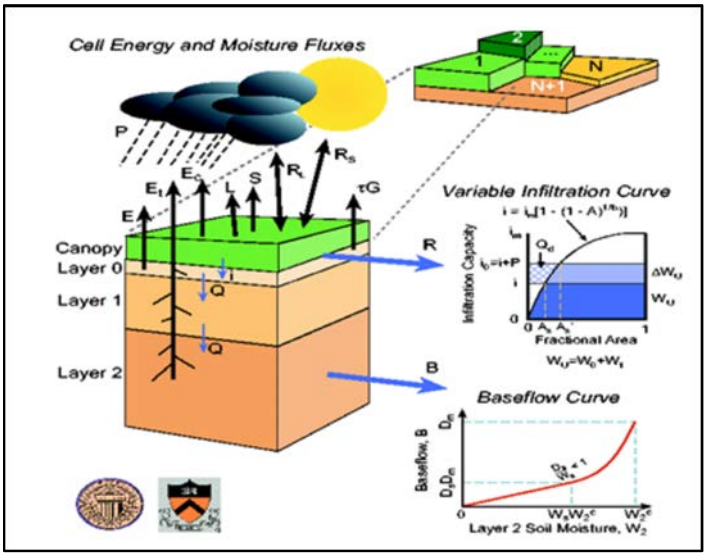



**Figure 1.3** Schematic of the VIC model adopted (Gao et al., 2009).

Scripts within the data pre-processing workflow are divided into four categories as shown in Figure 1.4. The scripts in the first category process the precipitation and the air temperature datasets. The scripts in the second category process the land surface datasets including topography, soil, and vegetation data. The scripts in third category process the wind speed dataset and the scripts in the last category create the final model input files for meteorological datasets. The datasets processed by the workflow are shown as ovals in the figure and include 1) meteorological forcing files (i.e., precipitation, wind, and minimum and maximum air temperature), 2) soil and vegetation parameter files, and 3) basin geospatial files. The primary inputs for the workflow are shown as parallelograms and include datasets from 1) the National Oceanic and Atmospheric Administration (NOAA) National Climatic Data Center (NCDC) (now the National Centers for Environmental Information (NCEI)), 2) the National Center for Atmospheric Research (NCAR) National Centers for Environmental Prediction (NCEP), 3) the National Aeronautics and Space

Administration (NASA) Land Data Assimilation System (LDAS) 4) the United States Geological Survey (USGS) HYDRO1K dataset, and 5) the PRISM Climate Group PRISM dataset. Each of these data providers has their own way of distributing data to users, and many of the datasets are large requiring methods for server-side processing to extract subsets of data for modeling studies.

### *1.1.3 Research Studies into Computational Reproducibility for Hydrologic Modeling*

In Chapter 2, I explore approaches for leveraging data grid technology in hydrologic modeling to support reproducible workflows using large datasets. The VIC model is used as a case study for this research. This is some of the first research applying data grid technology for hydrologic modeling. Its primary contribution is a general methodology for analyzing large, distributed data collections, by moving processing to data. This approach uses data grids to automate data transfers and staging, in combination with automated formal publication of generated data assets. This will be important as hydrologists seek to scale up watershed models to larger river basins where data sizes and computational processing make reproducibility more challenging.

In Chapter 3, I advanced the prior efforts for formalizing model metadata in hydrology by evaluating the OntoSoft Ontology as a means for structuring model metadata. The evaluation is performed using a data pre-processing workflow for the Variable Infiltration Capacity (VIC) hydrologic model that consists of multiple software components written by different scientists over time. The analysis begins by exploring what metadata hydrologists have already captured in unstructured forms. It then shows how this metadata could be organized into structured, machine-readable metadata using the OntoSoft Ontology. Therefore, a primary contribution of this study is an evaluation of the OntoSoft Ontology for describing software relevant to hydrologic modeling.

This is done by first understanding what metadata for hydrologic modeling software are already embedded in online resources, and then testing how this metadata maps to the OntoSoft Ontology.

In Chapter 4, I demonstrated how to integrate Hydroshare and GeoTrust easily and efficiently to reproduce model workflows. The MODLFOW groundwater model is used as an example to show the functionality provided by this cyberinfrastructure for creating open and reusable data analysis and model execution services. The primary contribution of this research is an end-to-end solution for converting scientific workflows to Docker containers, sharing these containers through an online collaborative environment, and executing these containers using cloud resources. Finally, Chapter 5 presents key conclusions across all three studies showing how reproducibility of computational hydrologic modeling was advanced through this research.

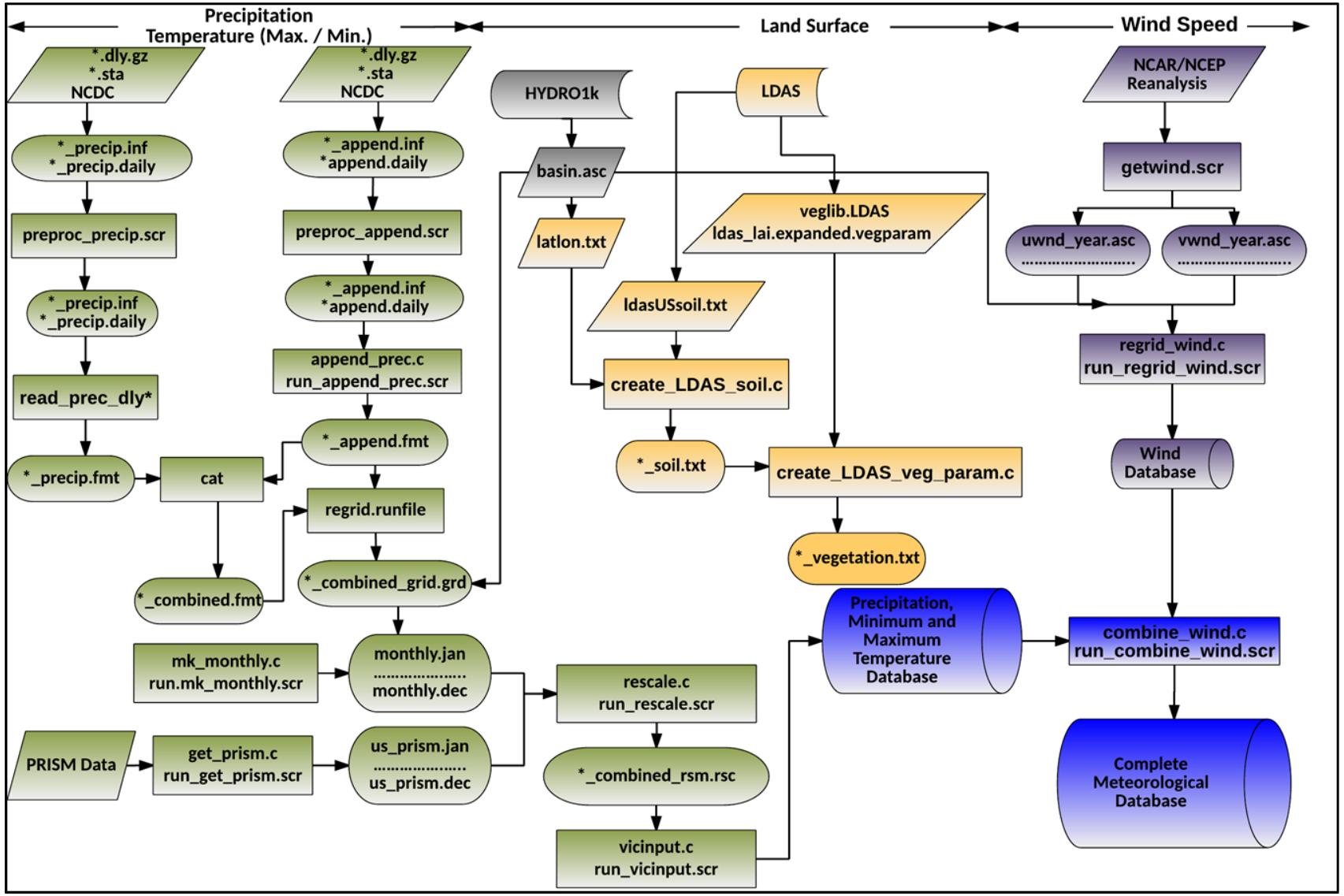



**Figure 1.4** Data pre-processing workflow for the VIC hydrologic model (adapted from Billah et al., 2016).

## 1.2 References


Billah, M. M., Goodall, J. L., Narayan, U., Essawy, B. T., Lakshmi, V., Rajasekar, A., & Moore, R. W. (2016). Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environmental Modelling {&} Software, 78, 31–39.

Gao, H., Tang, Q., Shi, X., Zhu, C., Bohn, T., & Su, F. (2009). Water Budget Record from Variable Infiltration Capacity (VIC) Model Algorithm Theoretical Basis Document (* this is a tentative author list) Department of Civil and Environmental Engineering University of Washington Seattle , WA 98195 Department of Civ, (Vic), 57pp. Retrieved from http://www.hydro.washington.edu/Lettenmaier/Models/VIC/Documentation/References.shtml

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 Resources Research, 50. https://doi.org/10.1002/ 2016WR019285

Liang, X., Lettenmaier, D. P., & Wood, E. F. (1996). One-dimensional statistical dynamic representation of subgrid spatial variability of precipitation in the two-layer variable infiltration capacity model. Journal of Geophysical Research: Atmospheres, 101(D16)(1), 21403–21422.

Liang, X., Wood, E. F., & Lettenmaier, D. P. (1996). Surface soil moisture parameterization of the VIC-2L model: Evaluation and modification. Global and Planetary Change, 13, 195–206. https://doi.org/10.1016/0921-8181(95)00046-1

Peng, R. D. (2011). Reproducible research in computational science. Science., 334(6060), 1226–1227.

Peng, R. D., & Eckel, S. P. (2009). Distributed reproducible research using cached computations. Computing in Science and Engineering, 11(1), 28–34. https://doi.org/10.1109/MCSE.2009.6

Singh, V. P., Asce, F., Woolhiser, D. A., & Asce, M. (2002). Mathematical Modeling of Watershed Hydrology. Journal of Hydrologic Engineering, 7(4), 270–292.

# Chapter 2: Server-Side Workflow Execution using Data Grid Technology for Reproducible Analyses of Data-Intensive Hydrologic Systems[1]

## 2.1 Introduction

There is an exponential growth in data available to geoscientists. The quantity of satellite data is growing rapidly (Acharya et al., 1998) and data from sensor networks are being widely used, in observatories such as the Critical Zone Observatory (CZO) (Anderson et al., 2008), the National Ecological Observatory Network (NEON) (Cowles et al., 2010), and the Ocean Observing Initiative (OOI) (Keller et al., 2008). Various groups are making available large collections of model-derived data including climate projections and reanalysis products for use by scientists. Public data repositories are used in many scientific disciplines as a means for sharing data collected by the so called "long-tail" of the scientific community (Dunlap et al., 2008). The number of public repositories will likely increase as funding agencies enforce requirements that scientists submit data products resulting from their funded research to these public repositories.

This exponential growth in data will impact modeling and data analysis approaches used in many geoscience disciplines. As datasets grow in complexity and resolution, there is a need for improved tools to derive information from raw data sources in support of a particular research objective. These challenges arise not only because processing large, semantically-unstructured datasets can be complex and time consuming, but also because capturing the computational workflows scientists complete for a particular study can be challenging. New strategies are needed

[1] This Chapter is a draft manuscript of a paper that has since been published. Readers are referred to the following citation for the final published version of the manuscript. Essawy, Bakinam T., Jonathan L. Goodall, Hao Xu, Arcot Rajasekar, James D. Myers, Tracy A. Kugler, Mirza M. Billah, Mary C. Whitton, and Reagan W. Moore. "Server-side workflow execution using data grid technology for reproducible analyses of data-intensive hydrologic systems." *Earth and Space Science* 3, no. 4 (2016): 163-175.

so that these scientist-authored computational workflows can make use of the latest available data and be reproduced and reused by other scientists.

One strategy for dealing with the growing volume of available data has focused on creating standards for accessing remote data collections using Web Service Application Programming Interfaces (APIs). The Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) Hydrologic Information System (HIS) has created standards for both an API called Water One Flow (WOF) and a data exchange language called Water Markup Language (WaterML) to facilitate transmission of hydrologic time-series data on large repositories using Web services (Maidment, 2008). The Open Data Access Protocol (OpenDAP) is another widely used protocol for accessing and sub setting scientific data using Web services (Cornillon et al., 2003). OpenDAP focuses in particular on gridded data and includes the concept of server-side data sub setting and format conversion that are essential for operating on large, remote files.

While the Web service approach for data access has significant benefits, it also has limitations in that the network protocol for performing the data transfers using Web services operates over HTTP. For large files, this approach is not optimal and potentially not feasible. Data grid technology provides an alternative approach for managing distributed data and computational resources. Data grids typically include features such as authentication, replication, authorization, auditing, and metadata support that are needed to manage large, distributed data collections (Foster, 2011; Rajasekar et al., 2010). These tools are better suitable for handling large files compared to Web services because they allow for parallel data transfers and provide automated fault tolerance and restarts when connectivity is lost during a transfer. Data grid technology has been used in the atmospheric and climate sciences, notably in the Earth System Grid and Earth System Grid Federation projects (Williams et al., 2011, 2008), but it has not been widely adopted

in other geosciences disciplines to date. In particular, research is needed to determine best practices and approaches for leveraging the technology to address specific needs in the hydrologic modeling community, which is the focus of this research.

The objective of this research is to explore approaches for leveraging data grid technology in hydrologic modeling to support reproducible workflows using large datasets. This is some of the first research applying data grid technology for hydrologic modeling. Its primary contribution is a general methodology for analyzing large, distributed data collections, by moving processing to data and using data grids to automate data transfers and staging, in combination with automated formal publication of generated data assets. This will be important as hydrologists seek to scale up watershed models to larger river basins where data sizes and computational processing make reproducibility more challenging.

The work is focused on a use case where a scientist wishes to create a workflow automating the data processing steps required to create a publication-ready figure from a large collection of model output files, greater than 2GB for a single run, produced using a Variable Infiltration Capacity (VIC) (Liang and Lettenmaier, 1994) hydrologic model. The use case, which is more fully explained in Section 3, demonstrates server-side data processing on large data collections, using data grid technology for data transfers, and federation with public data repositories for reproducibility of the analysis workflow. It represents one of the first applications of the newly developed Workflow Structured Object (WSO) functionality in the iRODs, which has general applicability to other scientific domains with significant data management challenges. While systems like MyExperiment (De Roure et al., 2009) also focus on server-side execution of scientist-authored workflows and provide advanced features for workflow sharing and publication, they focus on using Web services for data transfer rather than grid technology.

This research also addresses the challenge of federation across different cyberinfrastructure systems. It is likely that data-intensive studies will need to access many cyberinfrastructure systems for data gathering, processing, modeling, and publication. This study demonstrates this concept for a use case that involves three cyberinfrastructure systems: the DataNet Federation Consortium (DFC) for data storage and compute resources, the Sustainable Environment-Actionable Data (SEAD) for data publication, and Terra Populus (TerraPop) for data access. Federation across these systems requires agreed upon standards and protocols that allow for interoperability. Different types of federation are demonstrated in our solution in order to address the transfer and management of both large and small data collections.

This study is part of a special issue on the Geoscience Paper of the Future (GPF). GPF is envisioned as a paper where all digital assets used in the study are published as open, online resource published with unique identifiers and key metadata including titles, abstracts, licenses, authors, and contacts (Gil et al., In Review). In this study, the key digital assets are published through SEAD with Digital Object Identifiers (DOIs) and key metadata attributes. The research itself is also aimed at the vision and goals of GPF focusing in particular on the use case where computation is needed on distributed data resources. It seeks to define methods for moving data from distributed servers within a data grid automatically using federation approaches and defining workflows that aid in capturing the provenance of how data were moved and processed to create publication-ready visualizations generated using multiple reference data collections. As data volumes continue to grow, such techniques will be critical to achieve the GPF goals.

The remainder of the study is organized as follows. In Section 2 we provide background on data grid technology to orient the reader. In Section 3 we present the use case in further detail, followed by the design and implementation of a prototype system for solving the use case in

Section 4. Finally, we provide a discussion of key aspects of our approach in Section 5 before offering concluding remarks in Section 6.

## 2.2 Data Grid Technology

Data grids are systems that enable access and sharing of large data sets that are physically distributed across the Internet, but appear to the user as a single file management system. The Integrated Rule-Oriented Data System (iRODS) is a data management system that includes the capability to federate data grids (Rajasekar et al., 2010). Federation allows for the creation of virtual data collections by logically arranging data from distributed resources under a virtual collection hierarchy. Globus is another data grid technology and is used within scientific communities and includes GridFTP for fast data transfer of large files (Foster, 2011). While iRODS and Globus are commonly used within some specific scientific domains (Allcock et al., 2002; Kyriazis et al., 2008), their use is not widespread within the hydrology community.

Data grids are particularly useful for scientific communities such as hydrology that rely on multiple data and computational resource providers. The iRODS-powered Data Federation Consortium (DFC) grid, which is used for this research, was developed as part of a National Science Foundation (NSF) funded project and provides support for federation of both resources and services. The work reported here is part of the DFC project and uses a DFC data grid for storage and long-term access to datasets stored across heterogeneous resources. The core iRODS software is developed and maintained by the iRODS Consortium at the Renaissance Computing Institute (RENCI), which is a partnership between the University of North Carolina at Chapel Hill (UNC-CH) and the Data Intensive Cyber Environments (DICE) Center at UNC-CH. iRODS currently runs in Linux/Unix environments.

iRODS has a client-server architecture. The iRODS client software can be installed and run on any computer. Each iRODS grid installation has two types of servers: exactly one iRODS Metadata Catalogue (iCAT) server and one or more iRODS resource servers, most frequently storage resource servers, e.g., data disks. Our system was developed on iRODS release 4.0, which includes software for the iRODS client, the resource server, and the iCAT server. iRODS uses the term zone as an abstraction for the physical components of an iRODS grid installation, i.e., the iCAT server and one or more resource servers that are part of the grid.

This work uses the recent development of iRODS Workflow Structured Objects (WSO), which enable workflows to be executed directly with iRODS commands. While iRODS is a mature, widely used software tool, this is some of the first work using the WSO functionality of iRODS. Therefore, this research was completed as a close collaboration between hydrologists defining the scientific workflows and the iRODS and WSO developers made possible through the DFC project. One goal of this work was to provide an example use case of applying WSO that could be beneficial for other iRODS users with interests in utilizing WSO in the future.

Figure 2.1a provides an overview of the file structure for a WSO. A WSO requires two primary files: a workflow file (*.mss) and a parameter file (*.mpf). The workflow file defines the sequence of operations to be performed by the workflow and the parameter file lists the input arguments used when executing the WSO. The parameter file also specifies any files in iRODS that should be staged-in (transferred to the physical directory on the iRODS resource server where the WSO is executed) or staged-out (put into an iRODS collection) prior to and following the execution of the workflow (Rajasekar, 2014). Examples of workflow and parameter files are provided in iRODS documentation, specifically from

https://wiki.irods.org/index.php/Workflow_Objects_(WSO)#Files_in_WSO.

When the user creates and uploads a parameter file, iRODS automatically generates a run file (*.run), which is then used by the client to execute the workflow. One workflow file can be used to create many instances of a WSO with each instance having a unique parameter file (see the wso, wso0, and wso1 collections illustrated in Figure 1). The data files used by the workflow are stored in runDir collections. Within each WSO, there could be multiple runDir collections, one for each execution of the workflow. Workflows can include scripts and other scientist-authored code installed on the server in the iRODS/server/bin/cmd directory (Figure 2.1b).

A WSO is executed by performing the following steps. (1) The user issues the *iput* command, which is part of the iRODS icommands client library, to transfer a workflow file (*.mss) from a client machine into an iRODS collection. (2) The user issues the *imkdir* command to make a new collection within the collection containing the workflow file (see the wso collection shown in Figure 1). (3) The user issues the *imcoll* command to mount this newly created collection. (4) The user issues the *iput* command to transfer a parameter file (*.mpf) into the mounted collection. This operation results in the system creating a run file (*.run) in the mounted collection. (5) The user issues the *iget* command on the run file to execute the workflow. The system then creates a new collection in the mounted directory (see the runDir collection shown in Figure 1) and the staged-in and workflow generated output files are stored in this new collection. The same workflow can be executed for different parameter files by repeating steps 4 and 5 for a new parameter file, with each new parameter file resulting in an additional WSO collection (see wso0, wso1, …. shown in Figure 2.1) ("Workflow Objects (WSO)," 2013).

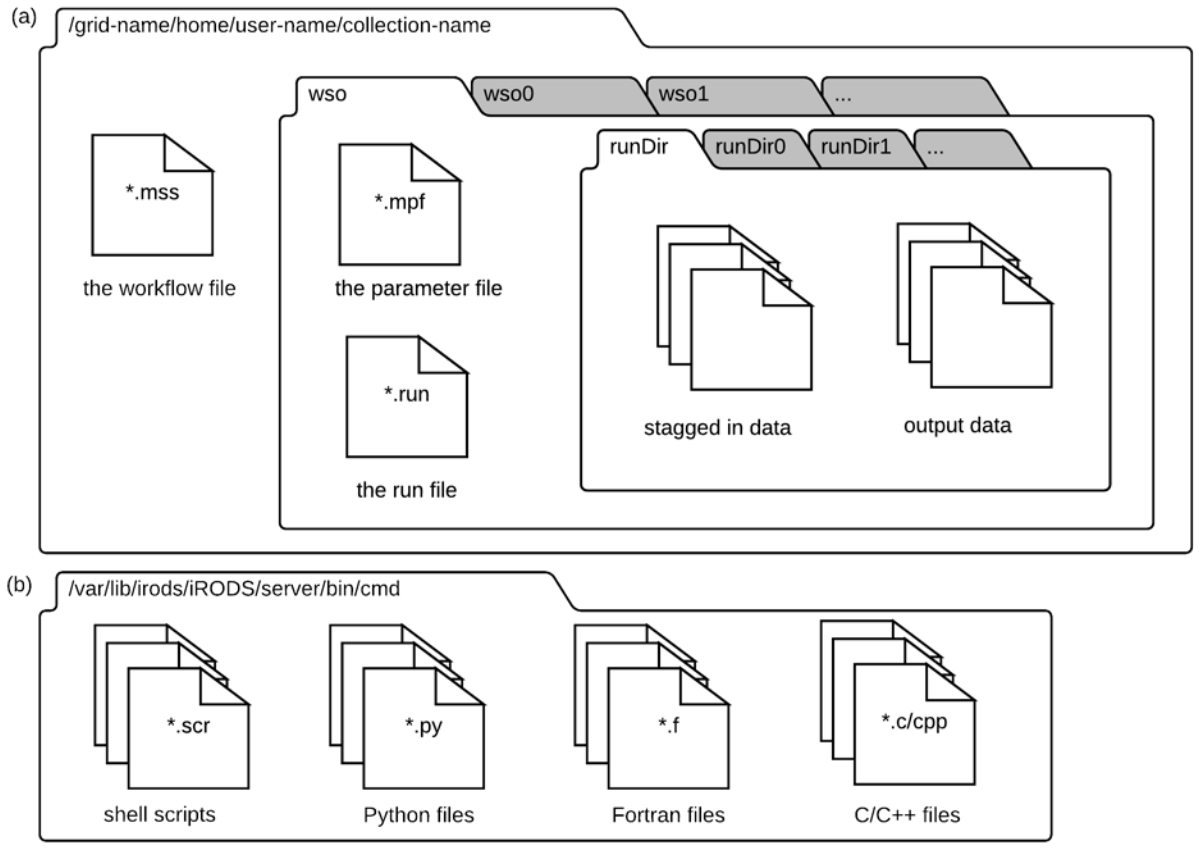



**Figure 2.1** (a) The structure of an iRODS Workflow Structured Object (WSO). (b) The WSO may utilize scripts installed in the iRODS/server bin/cmd directory for server-side data processing.

There are a number of workflow environments available to geoscientists, e.g., Kepler (Altintas et al., 2004), Taverna (Oinn et al., 2004), Triana (Harrison, Andrew, 2008), and Pegasus (Deelman et al., 2005). Like iRODS WSO, these workflow systems make trade-offs between power and flexibility. Many enable large-scale, parallel workflow execution on distributed resources, providing users real-time status information on the workflow execution (Vahi et al., 2013). While workflow systems share many similarities, there are also key differences, which can often be subtle, that determine their suitability for addressing particular use cases. We used iRODS WSO in this analysis because our use case required a data processing pipeline consisting of a set of scientist authored scripts that operate on data collections already within iRODS. Future work

comparing and contrasting iRODS WSO with other workflow environments for completing this or other use cases relevant to hydrologic modeling would be a useful extension to this research.

## 2.3 Use Case Description

The prototype software described in this study is designed to address a use case where a scientist has created a simulation using the Variable Infiltration Capacity (VIC) model for the Carolinas region of the United States. The model has been calibrated and validated for this region as part of a prior study (Billah et al., 2015) and can be used to address other hydrologic research questions as well. The scientist that created the model has published the model's input and output files on the Web for use by other scientists. A second scientist learns about the model and wishes to use the model's output files to test her own research question about drought impacts on counties within a study region. The scientist is interested in how soil moisture deficit predicted by the model varied for different populated communities within the study region. While this application is analyzing historical events, it would be relatively straight-forward to set up the calibrated model to analyze current conditions and to identify populated regions vulnerable to drought conditions within the region. Such information would be valuable to resource managers in better understanding the severity of the drought and its impact on population centers within the region.

The second scientist downloads the model output files published online by the first scientist and creates the visualization by writing her own Python scripts. The scientist downloads the population data for the study counties to a local working directory. The VIC soil moisture outputs are organized in a set of "flux files," one for each node in the modeling domain. The Python scripts sort through these data extracting relevant information and summarizing the soil moisture time series. Geospatial processing tools are used to relate the coordinates of the model nodes to counties in the study region. The result of this data processing is a comma separated values (CSV) file with

the soil moisture deficit and population for each of the five counties. Finally, the scientist programs the Python script to use this CSV file to produce a publication-ready figure for visualizing the drought impacts.

In addition to publishing the scripts and data files from this analysis on a public data repository, which is now a relatively straight-forward exercise given the proliferation of online data repositories, the scientist also wishes to publish the workflow used to perform the analysis as a Web executable resource. The scientist wishes to take this approach for the following reasons.

- Having the overall workflow be executable server-side means the scripts and model output data can be co-located, removing the need to download the large model output file to the scientist's machine prior to the workflow execution.

- By keeping datasets server-side, it is easier to ensure the data has not been modified after making a local copy (its provenance can be proven). With the ability to publish the model and reference data once, and to keep them on the server, only the visualization results need to be retrieved and published for subsequent runs.

- Having server-side execution of the workflow controls for potential variability across different hardware and software configurations on a client machine. Even with this relatively simple use case of creating a figure, there is potential for different operating systems and versions of analysis software to result in differences in the end product. These software dependencies could result in additional time for scientists to trouble shoot errors. More critically, these dependencies could result in an end product without errors or warnings, but with inconsistencies due to non-breaking differences between dependent software versions.

Simply put, having data and processing co-located on a server as a Web executable resource results in a more controlled environment, which is critical for reproducibility.

The scientist uses iRODS WSO to create the Web executable resource. As part of the WSO, the scientist defines the steps to automatically stage-in the required VIC output and population data that are stored in iRODS collections. The population data comes from TerraPop, which provides global-scale data sets that focus on human population characteristics, land use, land cover, and climate change (Minnesota Population Center, 2013). The Terra Populus data access system was used to create customized data extracts, combining variables from multiple sources into a single package. Users can browse the TerraPop collection and select the required variables; the variable required in this study was the total population for each county in the United States. After submitting our data request, the system generated a data package that included a shapefile for all the counties in the United States, with unique GEOID identifiers, and a CSV file that includes the GEOID and name of each county (Figure 2.2). This data package was then automatically uploaded onto the TerraPop grid as an iRODS collection. By federating the DFC-hydrology and TerraPop zones and configuring authorizations, we are able to have the population data remain on the TerraPop server and be automatically staged-in for use by the WSO.

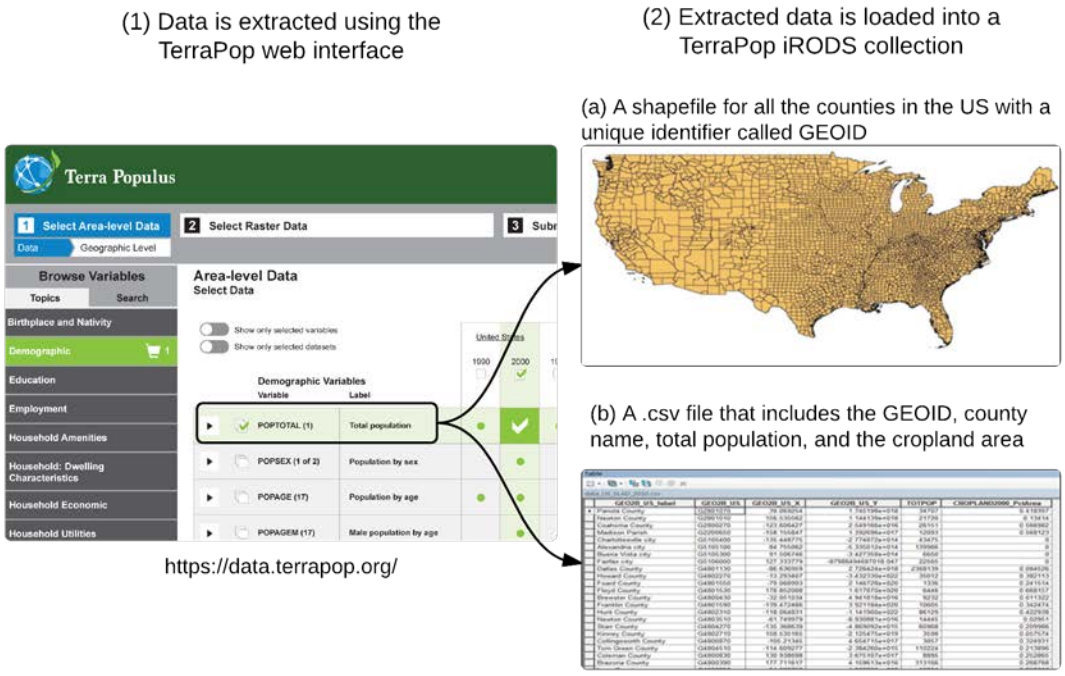



**Figure 2.2** Details on how the county-level population data is requested and extracted using the TerraPop web interface into an iRODS data collection. From this collection, iRODS stages-in the required files prior to the workflow execution.

Finally, the data (including code) resulting from the analysis are published using products provided by the Sustainable Environment-Actionable Data (SEAD) project (Myers et al., 2015). The SEAD project supports publication, preservation, and sharing of data generated by scientists including data generated by running models. Using SEAD, teams of researchers can upload, share, annotate, and review input datasets and model outputs within an access-controlled Project Space, and then formally publish collections of data with associated metadata and provenance for long-term preservation (generating a Digital Object Identifier (DOI) and standards-based archival package, and registering the data with the DataONE catalog for discovery). Our use of SEAD included manual entry of data and metadata via a web interface and bulk uploads of files and programmatic submission of the output figure with metadata to SEAD, which leveraged SEAD's RESTful Web API.

## 2.4 Prototype Software Design and Implementation

We present the prototype software aimed at addressing the use case by first describing the steps taken to configure the server-side software and data, next describing the steps required to configure the WSO, then describing the steps required to execute the WSO from the client machine, and concluding with a summary of the results from executing the workflow.

### *2.4.1 Server-Side Configuration*

To perform the server-side configuration, we first installed iRODS resource server version 4.0 software on an Elastic Cloud Computing (EC2) instance in the Amazon Web Services (AWS) cloud. We chose AWS because it provides on-demand computing resources and services that can be easily scaled to meet demands. The EC2 service provided through AWS allows users to rent virtual machines (instances) with different capabilities and pay by the CPU hour. For prototyping purposes, we used a Linux-based medium sized machine (m3) with 3.75 GB of memory, 4 vCPU, 15 GB of SSD-based local instance storage, and 64-bit platform for the iRODS resource server ("Amazon EC2 Instances," 2015). Next this new iRODS resource server was configured to be part of the DFC-hydrology zone that has its iRODS Metadata Catalog (iCAT) server on a machine running at RENCI. We had to configure the AWS EC2 instance to be associated with an elastic IP address to avoid having to update the EC2 instance's IP addresses in the iCAT server following each restart of the EC2 instance.

We then developed a WSO on the iRODS resource server to implement the data visualization workflow described in the use case. This required that the user have an account on the server itself with read/write access to the cmd directory (Figure 2.1b). It was also necessary to set read/execute rights on the files associated with the WSO so that they could be executed by the iRODS user account. We uploaded to the iRODS resource server the VIC model output files from

SEAD (where the original scientist had published them for use by the community), the Python scripts created by the scientist to generate the visualization, and the shell script, also created by the scientist, used to sequence the execution of the Python scripts on the iRODS resource server. The VIC source code is not included in SEAD because the source code is available from the developer's GitHub page instead (see https://github.com/UW-Hydro/VIC).

### *2.4.2 Client-Side Configuration*

The client machine can be any computer with the iRODS client software installed. In this prototyping work, we used a second EC2 instance as the client machine simply to avoid moving data into and out of the AWS cloud. We installed the icommands iRODS client software library on the client machine. The icommands software includes a set of commands that perform operations such as make a new directory (*imkdir*) or put a file into an iRODS collection (*iput*) (Weise et al., 2008). The icommands client library includes an environment configuration file that is used to point to a particular iRODS zone and set default user credentials for accessing the iRODS zone. In our case, we configured the icommands environment to operate on the DFC-hydrology zone and entered user credentials representing the scientist accessing the system.

The general file structure required for creating a WSO was described in Section 2 and in Figure 2.1a. For our particular application, we first created a workflow file (PopVsSm.mss) that specifies the steps required to execute the workflow. The workflow file simply specified that the workflow should execute the scientist-authored shell script installed on the iRODS server cmd directory. We put the PopVsSm.mss file into an iRODS collection and then made a new collection named "vic_soilmositure." We mounted this new collection, effectively making it a WSO.

### *2.4.3 Executing the Workflow*

Once the WSO is mounted, it is then possible to execute the workflow. This process is described in general in Section 2. Here we provide specifics of the WSO execution for the use case. The general flow of data and sequence of commands for executing the WSO execution for the use case is described in Figure 2.3.

(1) The user initiates execution of the workflow by issuing an *iget* command on the PopVsSm.run file that is in the mounted WSO collection. The PopVsSm.mpf parameter file defines the data required by the workflow and stages these files from different iRODS collections into the directory on the iRODS resource server where the WSO is executed. In our case, we staged-in the VIC model output data stored in the DFC-hydrology grid and county-level population data from the TerraPop grid. While these two datasets are stored within different grids, it is possible to gain access to the data directly using iRODS authentication because the grids are federated.

(2) Once all required data is staged into the iRODS resource server directory where the workflow is executed, the workflow file specifies that the scientist-authored shell script stored on the iRODS server should be executed. This shell script then calls a series of scientist-authored Python scripts that process the staged-in data to create the output figure.

(3) A final step in the shell script is publishing the figure resulting from the workflow automatically to a SEAD project space for sharing with colleagues and subsequent publication. The SEAD API is used for this purpose and allows for the submission of the file along with associated metadata to a SEAD project space.

(4) Upon completion of the workflow, key output data are staged-out into iRODS collections according to specifications in the parameter file. This allows the files to be accessible to authorized users in the grid.

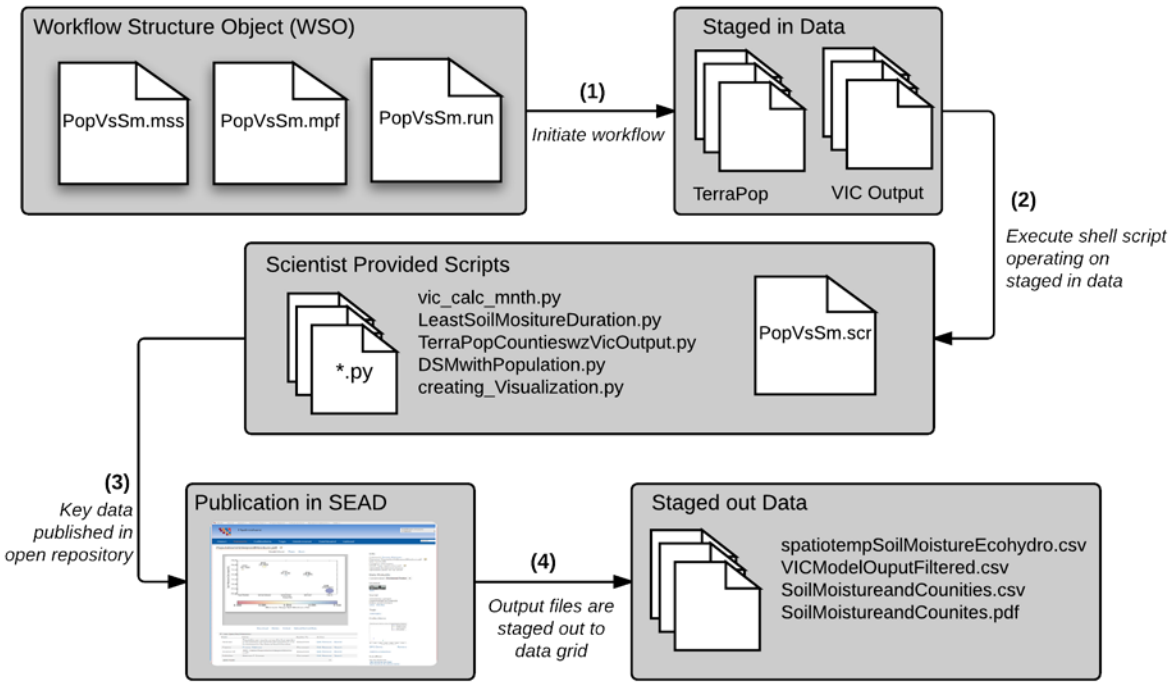



**Figure 2.3** The steps that occur on the server-side when a user executes the WSO. Data is staged-in from iRODS collections, scientist-authored scripts are run to create the figure, data is published through a SEAD project space using the SEAD API, and key output data is staged-out back into iRODS collections.

Figure 2.4 shows the steps for executing a WSO from a user's perspective when working with the icommands client library. The user must know which iRODS collection contains the script files required for executing the WSO to be able to execute it. Once the user has logged into the client machine, the user changes the working directory to the iRODS logical path where the WSO has been mounted. In this case, the WSO was mounted as the "vic_soilmoisture" collection. The user next issues an *iput* command to put the parameter file (PopVsSm.mpf) into the mounted WSO. This step is not illustrated in Figure 2.3 for brevity, but results in the generation of a run file (popvssm.run) in the collection. Finally, the client executes the workflow by issuing an *iget* command on the popvssm.run file.

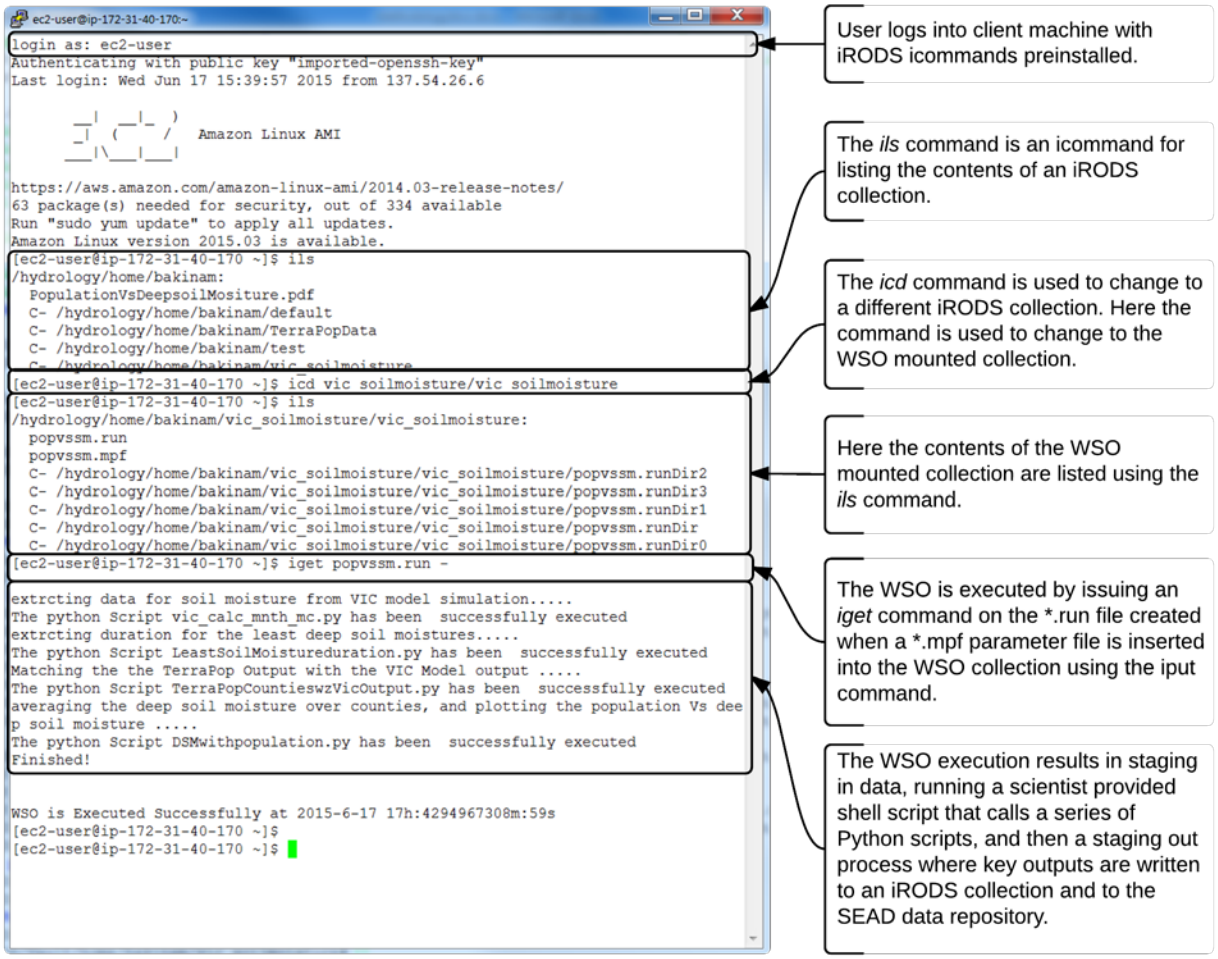



**Figure 2.4** The steps required from a client machine in order to execute the WSO using the icommands client library.

### *2.4.4 Results from the Workflow Execution*

When the workflow is executed, the output messages are written to the console, although all computation is performed on the server-side and no data (other than the output messages) are transferred to the client machine. Once the workflow execution has completed, the user can access the output collection called runDir resulting from the workflow execution. The runDir file contains by default the stdout from the execution of the workflow along with any staged-in and derived data from the workflow ("Workflow Objects (WSO)," 2013).

The workflow also results in publication of the workflow results to a SEAD project space. Figure 2.5 shows the data collections as they appear through the SEAD project space website. Most data were uploaded using the SEAD web interface. Figure 2.6 shows the figure resulting from the WSO execution that was automatically written to the SEAD project space using the SEAD API as a final step in the WSO execution.

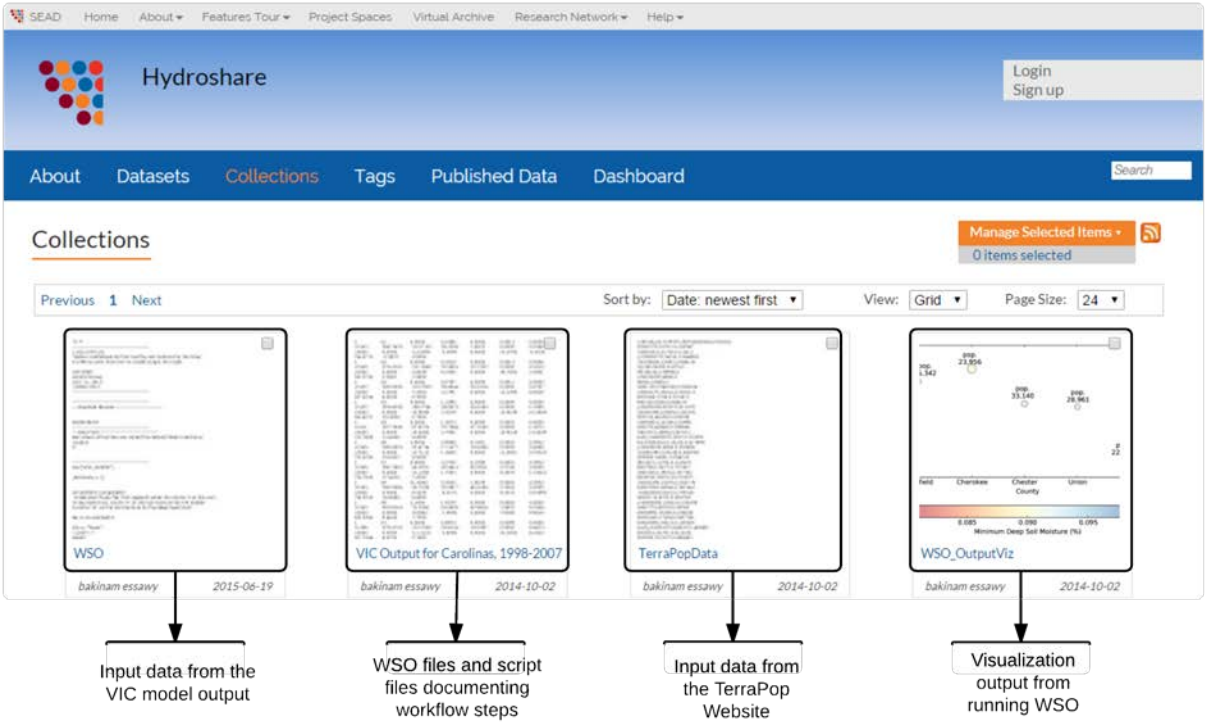



**Figure 2.5** Contents of the Sustainable Environment Actionable Data (SEAD) project space used for storing and accessing data used in the workflow.

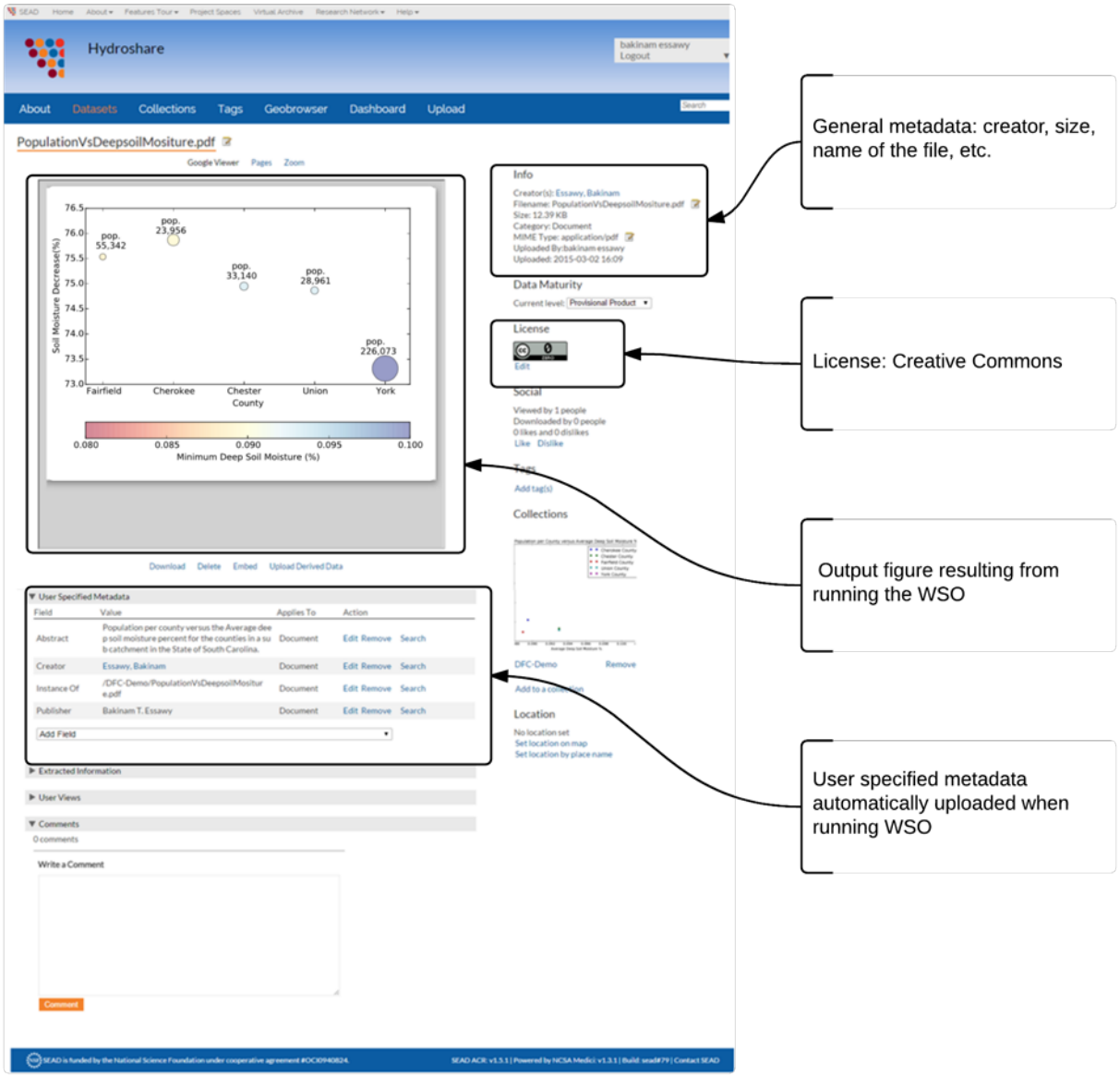



**Figure 2.6** View of figure, produced by executing the WSO, within the SEAD project space. The workflow uses the SEAD API to upload this resource along with metadata to the SEAD project space.

## 2.5 Discussion

### *2.5.1 Reproducibility*

To support transparency and reproducibility of this work as envisioned by the Geoscience Paper of the Future (GPF) project, the data collections in the use case (e.g., the VIC output files, the TerraPop data, the WSO files, and the output figure) were published in SEAD. As part of this publication process, each collection was given metadata including a brief abstract, creators, the publisher, and then published to generate a Digital Object Identifier (DOI) Table 2.1. The output figure resulting from the WSO execution was first written to a SEAD project space along with basic metadata as a final step in the WSO execution using the SEAD API. From there, the scientist logged into the SEAD web interface and set additional metadata fields to publish the resource with an assigned DOI. Any combination of automated and manual entry is supported and researchers can choose which data to publish. In our case, we automatically captured outputs from multiple test runs before manually selecting, annotating, and publishing (including creating a DOI for) only the final run.

Use of an open, metadata-aware repository makes it simple to capture additional derived data and provenance information as research continues. By publishing the reference data, scripts, and output data separately in SEAD, we also demonstrate the ability for larger reference data to be published once, and then referenced via provenance links from the derived output files that could be generated by many researchers over time. For example, the VIC output files used in this workflow may be used in other research studies. If each publication using these VIC output files references its DOI, it will be possible to track the impact of the model output files through citation counts similar to what is done now for tracking citation counts of research papers.

Other end-points could be used for publishing key digital assets from the WSO workflows. For example, the Consortium of Universities for the Advancement of Hydrologic Sciences, Inc. (CUAHSI) HydroShare system is in development and could serve as an alternative or secondary end-point for publishing results with more discipline-specific metadata (Horsburgh et al., 2015; Morsy et al., 2014; Tarboton et al., 2014), as could systems such as FigShare or Zenodo. We anticipate a growing number of such repositories and for federation between them (e.g., SEAD is already a member node in DataOne (Michener et al., 2012), advertising our WSO publications through DataONE's catalog). This research shows how iRODS WSO could play an important role in moving data resources within such data repositories to and from computational resources to support data computation use cases.

**Table 2.1** Key digital assets used in the study that are published through SEAD with basic metadata.

| Title | DOI | Author | Contact | Abstract | License |
|---|---|---|---|---|---|
| TerraPop Data Extract | 10.5967/M08P5XH5 | Essawy, Bakinam | Goodall, Jonathan | Population data extracted from TerraPop (https://data.terrapop.org) for the study region. | Creative Commons (CC) |
| VIC Output for Carolina, 1998-2007 | 10.5967/M0DF6P6F | Essawy, Bakinam | Goodall, Jonathan | Output from a VIC model for the Carolinas, USA calibrated for the period 1998-2007 to study drought impacts. | Creative Commons (CC) |
| WSO | 10.5967/M0J67DXR | Essawy, Bakinam | Goodall, Jonathan | The scripts and related files used to create the iRODS Workflow Structured Object (WSO). | Creative Commons (CC) |
| WSO_OutputViz | 10.5967/M0513W51 | Essawy, Bakinam | Goodall, Jonathan | Impact of 2007 drought on five counties in the study region. | Creative Commons (CC) |

Using a public cloud offers further opportunities for reproducibility. It is possible to quickly set up virtual machines (VMs) with a variety of operating systems to reproduce computational analyses. It is also possible to capture images of VM instances that can be stored for future reproducibility. Exploring the use of virtual containers (e.g., the Docker project) rather than VMs would be a useful extension to this work. Virtual containers can reduce set up time and storage costs compared to VMs for software, like what was used in this work, that run in a Linux operating system.

### *2.5.2 Federation*

Federation across cyberinfrastructure systems is a key aspect of this work. Federation describes how distinct and formally disconnected systems interoperate. There is a growing set of cyberinfrastructure systems available to scientists, and many studies will benefit from the use of more than one of these systems. Effective ways for federating across these systems will result in powerful tools that save scientists' time and encourage reproducibility through automatic data transfers handled directly by systems. This concept was illustrated in our study by showing how distinct cyberinfrastructure systems can be federated and used collectively within a single workflow execution.

Figure 2.7 provides a depiction of the workflow that emphasizes different data collections and approaches for federating between DFC, TerraPop, and SEAD. The use case in this study represents two levels of federation that we believe are relevant for most scientific studies. The federation between the AWS machine where the workflow was executed and the TerraPop reference data is what we term a strong federation, while the federation between the AWS machine and SEAD is what we term a weak federation. A strong federation is based on a strong trust model where one data grid administrator can add credentials of users of other data grid, and grant access

to resources based on authentication through other data grids. One primary benefit of this level of federation is that data grid technology can be used to transfer files between the two systems. For large files, this level of federation will be important because of the functionality provided by data grids like iRODS that are designed specifically to ensure rapid and successful transfer of large files over a network. Weak federation, based on federation through Web service APIs, allows for greater flexibility and less required trust between systems, because all operations are through services. Transferring large data through Web services, however, it not ideal for the reasons we outlined in Section 2.

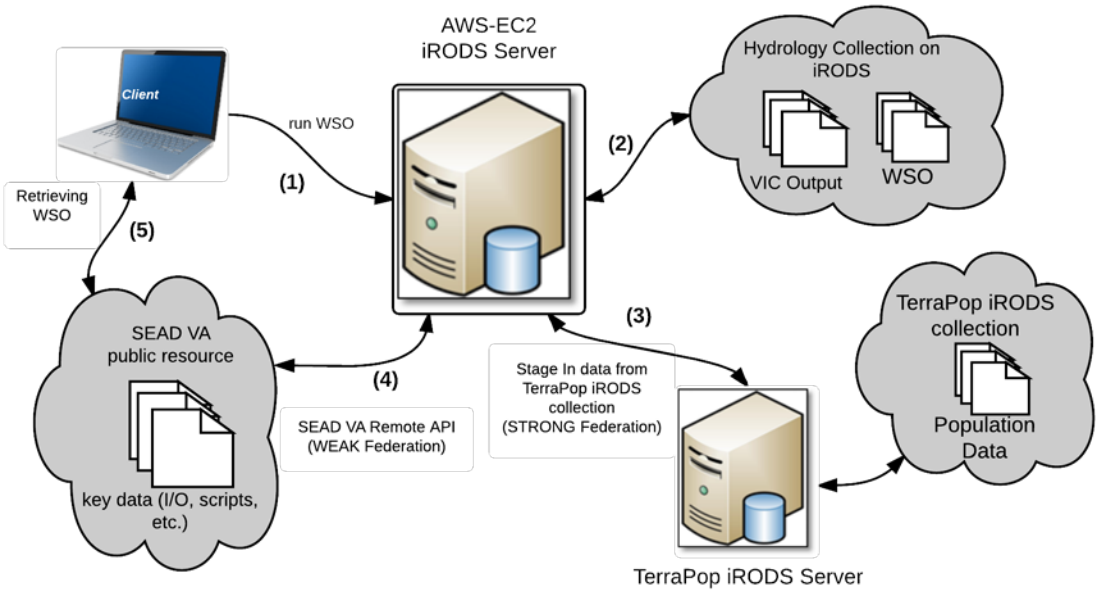



**Figure 2.7** Main components and data flow in the workflow emphasizing data collections and federation approaches

### *2.5.3 Adoption*

While there are many advantages to the approach described in this study, there are also important barriers to adoption, especially in terms of the current prototype system. Currently, users of the system need to be familiar with an iRODS client (e.g., the icommands client library used in this study). They must also be aware of steps for executing a WSO. Developers need an

understanding of how to structure new WSOs and will need access to the server running the iRODS resource server software for installation and configuration of the WSO.

There are opportunities for abstracting the complexity of directly interfacing with iRODS WSO for end users in order to encourage broader adoption of the technology. One way to do this would be to have someone familiar with iRODS WSO take input from the scientist including the scripts needed to execute the workflow and the location (iRODS logical path name) of the input data for the scripts. The administrator would then mount a WSO with an example parameter file and make it available through the system to end-users. The user could then execute the workflow either using the icommands client library, as described in the study, or through other tailored client applications able to operate on iRODS collections including executing WSOs stored within iRODS collections. We believe this would be a fairly straightforward process for moving scientist-authored codes into a form that is Web-executable.

### *2.5.4 Data Size and Heterogeneity Challenges*

This work only begins to illustrate the potential benefit of using data grid technology for executing workflows that require heterogeneous data from distributed data sources. We showed how WSOs allow for automatically staging-in of required data distributed across a data grid. We also showed how data produced from the workflow can be staged-out, meaning written to collections in the data grid where it can be accessible to other users. While it was not demonstrated in this use case, one can execute a distributed workflow across the network on multiple iRODS resource server using WSO.

This approach allows the location of the input and output files for a computational tool to be independent of the location where the processing is done. However, unlike approaches that rely only on Web service APIs for data staging prior to workflow execution, iRODS provides a more

robust data staging approach that leverages grid technology. While the use case demonstrated the concept using fairly small file sizes, the solution we used can be applied to larger terabyte scale data as well. Given that modeling in many geoscience disciplines requires access to large, distributed data, data grid technology provides a powerful way for data staging associated with workflow execution.

## 2.6 Conclusions

The focus of this study is on creating scientist-authored workflows as Web-executable resources in data grids. The iRODS WSO provides researchers with the ability to publish their research methods for computational studies as workflows that specify the tools, data, and sequence of steps taken to complete the study. All of these digital objects (data, software, model outputs, etc.) can be made accessible to other users of the data grid as well as to non-grid users through publication in SEAD.

There are many challenges in reaching the ultimate goal of reproducibility, especially when dealing with data-intensive modeling analyses that require a large, diverse set of input data and generate a large, diverse set of output data. Through this work, we argue that reproducibility will require more server-side data processing, where reference data is managed along with the model itself, than what is common now. This is due to the large and increasing size of datasets used by geoscientists, and the growing complexity of software and software dependencies that require constrained environments to ensure reproducibility.

We also argue for multiple federation approaches as means for providing interoperability across the variety of cyberinfrastructure systems needed for data access, analysis, modeling, and publication services. Federation approaches most often used in geoscience disciplines emphasize Web service APIs, however to support large datasets, the community should have broader adoption

of data grid federation approaches as well. The use of both approaches was demonstrated for a use case that leveraged four federated but heterogeneous cyberinfrastructure systems: DFC, TerraPop, and SEAD, and via an existing connection with SEAD, DataONE.

Any approach for making scientific computations into Web-executable resources must have a low barrier to entry for users. We have proposed an approach that allows scientists to write scripts as is typically done now for data analysis using languages familiar to scientists, and then making these scripts available as Web-executable resources to scientists using iRODS WSO technology. Future work should explore embedding of iRODS WSOs into systems that include tailored interfaces for scientific communities. Then, rather than the steps described in the study for executing WSO that include the use of the icommand client library, the end user could have a more tailored interface for viewing and executing workflows that abstracts technical details from the end user.

There are encouraging trends toward increased publication of data (including code) used in scientific studies. It is important that the momentum behind these trends result in scripts and workflows as Web-executable resources to capture their full potential in advancing reproducibility goals. The advantages of Web executable resources include the increased ability to share, reproduce, and collaborate on scientists-authored workflows. While the potential of scientific scripts and workflows as Web executable resources is clear, important issues remain related to managing large data and computation collections. We have demonstrated here an approach using data grids for addressing this challenge, and have argued for moving processing to reference data stored within data grids as a method for creating reproducible scientific workflows on large datasets.

## 2.7 References


Acharya, A., Uysal, M., Saltz, J., 1998. Active disks: programming model, algorithms and evaluation. SIGPLAN Not. 33, 81–91. doi:10.1145/291006.291026

Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S., 2002. Data management and transfer in high-performance computational grid environments. Parallel Comput. 28, 749–771. doi:10.1016/S0167-8191(02)00094-7

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S., 2004. Kepler : An Extensible System for Design and Execution of Scientific Workflows, in: 16th International Conference On. IEEE, 2004. pp. pp. 423–424. doi:10.1109/SSDM.2004.1311241

Amazon EC2 Instances [WWW Document], 2015. URL http://aws.amazon.com/ec2/instance-types/ (accessed 6.7.15).

Anderson, S.P., Bales, R.C., Duffy, C.J., 2008. Critical Zone Observatories: Building a network to advance interdisciplinary study of Earth surface processes. Mineral. Mag. 72, 7–10. doi:10.1180/minmag.2008.072.1.7

Billah, M.M., Goodall, J.L., Narayan, U., Reager, J.T., Lakshmi, V., Famiglietti, J.S., 2015. A methodology for evaluating evapotranspiration estimates at the watershed-scale using GRACE. J. Hydrol. 523, 574–586. doi:10.1016/j.jhydrol.2015.01.066

Cornillon, P., Gallagher, J., Sgouros, T., 2003. OPeNDAP: Accessing data in a distributed, heterogeneous environment. Data Sci. J. 2, 164–174. doi:10.2481/dsj.2.164

Cowles, T., Delaney, J., Orcutt, J., Weller, R., 2010. The Ocean Observatories Initiative: Sustained Ocean Observing Across a Range of Spatial Scales. Mar. Technol. Soc. 44(6), 54–64. doi:http://dx.doi.org/10.4031/MTSJ.44.6.21

De Roure, D., Goble, C., Stevens, R., 2009. The design and realisation of the Virtual Research Environment for social sharing of workflows. Futur. Gener. Comput. Syst. 25, 561–567. doi:10.1016/j.future.2008.06.010

Deelman, E., Singh, G., Su, M., Blythe, J., Gil, Y., Kesselman, C., 2005. Pegasus : A framework for mapping complex scientific workflows onto distributed systems. Sci. Program. 13, 219–237.

Dunlap, R., Mark, L., Rugaber, S., Balaji, V., Chastang, J., Cinquini, L., DeLuca, C., Middleton, D., Murphy, S., 2008. Earth system curator: metadata infrastructure for climate modeling. Earth Sci. Informatics 1, 131–149. doi:10.1007/s12145-008-0016-1

Foster, I., 2011. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. IEEE Comput. Soc. 15, 70–73. doi:doi:10.1109/MIC.2011.64

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., In Review. Towards the Geoscience Paper of the Future: Best Practices for Documenting and Sharing Research from Data to Software to Provenance. Earth Sp. Sci.

Harrison, Andrew, et al, 2008. WS-RF workflow in Triana. Int. J. High Perform. Comput. Appl.

22, 268–283. doi:10.1177/1094342007086226

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M.J., Tarboton, D.G., 2015. Hydroshare: Sharing Diverse Environmental Data Types and Models as Social Objects with Application to the Hydrology Domain. JAWRA J. Am. Water Resour. Assoc. n/a–n/a. doi:10.1111/1752-1688.12363

Introduction to Workflow as Objects [WWW Document], 2012. URL https://wiki.irods.org/index.php/Introduction_to_Workflow_as_Objects (accessed 6.7.2015).

Keller, M., Schimel, D.S., Hargrove, W.W., Hoffman, F.M., 2008. A continental strategy for the National Ecological Observatory Network. Front. Ecol. Environ. 6, 282–284. doi:10.1890/1540-9295(2008)6[282:ACSFTN]2.0.CO;2

Kyriazis, D., Tserpes, K., Kousiouris, G., Menychtas, A., Varvarigou, T., 2008. Data Aggregation and Analysis : A Grid-based approach for Medicine and Biology. Int. Symp. on. IEEE 841–848.

Liang, X., Lettenmaier, D.P., 1994. A simple hydrologically based model of land surface water and energy fluxes for general circulation models. J. Geophys. Res. 99, 14,415–14,428. doi:10.1029/94JD00483

Maidment, D.R., 2008. Bringing Water Data Together. J. Water Resour. Plan. Manag. 134, 95–96.

Michener, W.K., Allard, S., Budden, A., Cook, R.B., Douglass, K., Frame, M., Kelling, S., Koskela, R., Tenopir, C., Vieglais, D.A., 2012. Participatory design of DataONE—Enabling cyberinfrastructure for the biological and environmental sciences. Ecol. Inform. 11, 5–15. doi:10.1016/j.ecoinf.2011.08.007

Minnesota Population Center, 2013. Terra Populus: Beta Version [Machine-readable database]. Minneapolis: University of Minnesota.

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

Myers, J., Hedstrom, M., Akmon, D., Payette, S., Plale, B.A., Kouper, I., McCaulay, S., McDonald, R., Suriarachchi, I., Varadharaju, A., Kumar, P., Elag, M., Lee, J., Kooper, R., Marini, L., 2015. Towards Sustainable Curation and Preservation: The SEAD Project's Data Services Approach. Proc. IEEE 11th Int. e-Science Conf. Munich, Ger. doi:10.1109/eScience.2015.56

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P., 2004. Taverna : a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20, 3045–3054. doi:10.1093/bioinformatics/bth361

Rajasekar, A., 2014. Workflows [WWW Document]. 6th Annu. iRODS User Gr. Meet. June 2014 Inst. Quant. Soc. Sci. MA. URL http://irods.org/wp-content/uploads/2014/06/Workflows-iRUGM-2014.pdf (accessed 8.12.15).

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

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

Vahi, K., Harvey, I., Samak, T., Gunter, D., Evans, K., Rogers, D., Taylor, I., Goode, M., Silva, F., Al-shakarchi, E., Mehta, G., Jones, A., Deelman, E., 2013. A General Approach to Real-time Workflow Monitoring, in: A General Approach to Real-Time Workflow Monitoring. In High Performance Computing, Networking, Storage and Analysis (SCC). pp. 108–118. doi:10.1109/SC.Companion.2012.26

Weise, A., Wan, M., Schroeder, W., Hasan, A., 2008. Managing Groups of Files in a Rule Oriented Data Management System ( iRODS ). Comput. Sci. 321–330. doi:10.1007/978-3-540-69389-5_37

Williams, D.N., Ananthakrishnan, R., Bernholdt, D.E., Bharathi, S., Brown, D., Chen, M., Chervenak, A.L., Cinquini, L., Drach, R., Foster, I.T., Fox, P., Hankin, S., Henson, V.E., Jones, P., Middleton, D.E., Schwidder, J., Schweitzer, R., Schuler, R., Shoshani, A., Siebenlist, F., Sim, A., Strand, W.G., Wilhelmi, N., Su, M., 2008. Data management and analysis for the Earth System Grid. J. Phys. Conf. Ser. 125, 012072. doi:10.1088/1742-6596/125/1/012072

Williams, D.N., Lawrence, B. N. Lautenschlager, M. Middleton, D., Balaji, V., 2011. The Earth System Grid Federation: Delivering globally accessible petascale data for CMIP5, in: Proceedings of the 32nd Asia-Pacific Advanced Network Meeting. pp. 121–130. doi:10.7125/APAN.32.15

Workflow Objects (WSO) [WWW Document], 2013. URL https://wiki.irods.org/index.php/Workflow_Objects_(WSO) (accessed 6.7.15).

# Chapter 3: Evaluation of the OntoSoft Ontology for Describing Metadata for Legacy Hydrologic Modeling Software[2]

## 3.1 Introduction

Hydrologists use many different computational models, with each model tailored to address specific questions and problems. Hydrological modeling has a long history, and many computational models have decades of development effort and many model versions behind them (Singh et al., 2002). In many cases, there has been splintering of the model code base where the original model code has started to be developed along different paths (e.g., MODFLOW). This causes confusion as to which specific version of software was used for a given modeling application. Further complicating the issue, models often have supporting software beyond the physical process-representations within the model engine itself. This software is used to create input datasets for the model (i.e., data pre-processing) and to analyze or visualize the output from the model (i.e., data post-processing). Organizing and categorizing this broad collection of modeling software so that it is possible to uniquely identify the software used to perform a study is a significant challenge.

The need to better manage the growing volume of software used for hydrologic modeling is central to the larger challenge of computational reproducibility. The common approach for achieving reproducibility has been for researchers to provide sufficient detail within a journal paper's methods section to allow for reproducing the study's results. Growing complexity in computational analyses means this approach is no longer sufficient. Scientific disciplines are trying different approaches to address this problem including model repositories, documentation, on-line

[2] This Chapter is a draft manuscript of a paper that has since been published. Readers are referred to the following citation for the final published version of the manuscript. Essawy, Bakinam T., Jonathan L. Goodall, Hao Xu, and Yolanda Gil. "Evaluation of the OntoSoft Ontology for describing metadata for legacy hydrologic modeling software." *Environmental Modelling & Software* 92 (2017): 317-329.

model execution, and scientific workflows (De Roure et al., 2009; Essawy et al., 2016; JB et al., 2007; Lud et al., 2006; Roure et al., 2010). One of the main purposes of these approaches is to make models easier to reuse so that scientists can advance the model while achieving reproducibility and strengthening the decisions based upon these models (Cassey and Blackburn, 2006; Hutton et al., 2016; Scholten et al., 2000).

To achieve "reproducible software" (Peng, 2011) for hydrologic modeling, not only does the software and data need to be shared, but also their associated metadata. Metadata is structured information for describing and explaining a digital resource that makes it easier to manage, retrieve, and use that resource (NISO, 2004). Metadata is now a common term for describing data sets, but metadata is less commonly used for describing software. Software for data collection, storage, retrieval, processing, and management has improved greatly, and has significantly contributed to the development of comprehensive distributed hydrological models (Singh et al., 2002). Capturing metadata for hydrologic modeling software is one of the steps required to make the software reproducible (Higgins, 2007; Mcdougal et al., 2016). Little attention has been paid to metadata for describing these software advances. Computational reproducibility also requires other advanced uses of standard software practices beyond metadata tools including version control, strong commenting and documentation, and code modularity.

The limited past efforts to define metadata for hydrologic models have largely focused on describing well maintained and widely used hydrologic models as a single information resource. Like data, however, there is a long-tail of software used to perform and support hydrologic modeling (Heidorn, 2008). Models are often the combination of smaller software modules or components contributed over time by a large number of individuals and groups. Taking a more granular view of models by diving into the details of the software provenance and attempting to

capture this provenance using metadata is necessary for many reasons. Some of these reasons include 1) providing attribution for software contributions, 2) maintaining and archiving existing models, 3) providing information that aids in installing and executing models, and 4) ultimately fostering reproducibility.

Metadata for hydrologic models is being collected and recorded, but it is unstructured, informal and distributed. The available metadata for these models are scattered across model documentation, source code repositories, model publication repositories, user forums, and other publically available resources. Metadata such as who created the model, when the model was created, and the type of input and output data for the model can be found from these sources for many scientific models, but are provided in human-readable form. Not having this information in a machine-readable form limits its utility and does not scale well to the growing volume of scientific software. Metadata needs to be in machine readable formats to be most useful (e.g. RDF, XML).

Efforts to establish more formalized, machine-readable formats for hydrologic model metadata include efforts through the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) HydroShare project and the Community Surface Dynamics Modeling System (CSDMS) project. HydroShare describes metadata for two key modeling concepts: a model program and a model instance. The model program is the software for executing the model and the model instance is the input files required for executing the model (Horsburgh et al., 2015; Morsy et al., 2014a; Tarboton et al., 2014b). A metadata framework has been proposed for both of these concepts that extend the Dublin Core Metadata Standard. The CSDMS project created a catalog of model programs across the surface dynamics community, which includes

hydrology, and captured metadata for these model programs (Peckham et al., 2013; Peckham and Goodall, 2013).

Recent related activities have focused on designing standard metadata for describing software with a particular focus on scientific software. OntoSoft is a project that is part of the National Science Foundation EarthCube Initiative and provides an ontology and portal for addressing the challenge of capturing metadata for scientific software in a formal way (Gil et al., 2016b, 2015). The metadata captured by the OntoSoft Ontology focuses on the knowledge needed for software sharing and reuse (Ratnakar and Gil, 2015). It is recommended for documenting software in scientific papers that follow best practices for reproducible research, open science, and digital scholarship (Cédric H David et al., 2016; Gil et al., 2016a), and has been used to document scientific software in published articles, e.g., (Fulweiler et al., 2016; Pope, 2016; Yu et al., 2016). OntoSoft is used in the research reported in this study because it was designed and developed by experts in the semantic metadata community, in contrast to past efforts for hydrologic model metadata that was designed and developed by hydrologists. An underlying question that the research reported in this study begins to address is whether this more general scientific metadata ontology is appropriate and useful for describing hydrologic modeling software.

The objective of this study is to advance prior efforts for formalizing model metadata in hydrology by evaluating the OntoSoft Ontology as a means for structuring model metadata. The evaluation is performed using a data pre-processing workflow for the Variable Infiltration Capacity (VIC) hydrologic model that consists of multiple software components written by different individuals over time. The VIC model is used by large community; over 500 publications used this model since 1993. The analysis begins by exploring what metadata hydrologists here already captured in unstructured forms. It then shows how this metadata could be organized into

structured, machine-readable metadata using OntoSoft Ontology. Therefore, the primary contribution of this work is an evaluation of the OntoSoft Ontology for describing software relevant to hydrologic modeling. This is done by first understanding what metadata for hydrologic modeling software are already embedded in online resources, and then testing how this metadata maps to the OntoSoft Ontology.

## 3.2 Background

### *3.2.1 Variable Infiltration Capacity (VIC) Model Pre-processing Workflow*

VIC is a macro scale hydrologic model that applies water and energy balances to simulate terrestrial hydrology at a regional spatial scale (Liang et al., 1996b). Like many hydrologic models, the VIC model requires significant effort to prepare its input data. Figure 3.1 shows the data processing workflow used to generate the meteorological and land surface input datasets for a VIC model simulation. This workflow consists of a sequence of 15 data processing steps, each step requiring input datasets from different sources, and many of the datasets having unique data models (Billah et al., 2016b). These scripts are written with different programming languages including Fortran 77, C, and C++. Shell scripts are used throughout the workflow to execute these steps and perform other commands required to complete the data processing tasks.

The workflow is divided into four categories as shown in Figure 3.1. The first category of scripts process the precipitation and the air temperature datasets, the second category of scripts process the land surface datasets including topography, soil, and vegetation data, the third category of scripts process the wind speed dataset, and the last category of scripts create the final model input files for meteorological datasets. The datasets processed by the workflow are shown as ovals and include 1) meteorological forcing files (i.e., precipitation, wind, and minimum and maximum air temperature), 2) soil and vegetation parameter files, and 3) basin geospatial files. The primary

inputs for the workflow are shown as parallelograms and include datasets from 1) the National Oceanic and Atmospheric Administration (NOAA) National Climatic Data Center (NCDC) (now the National Centers for Environmental Information (NCEI)), 2) the National Center for Atmospheric Research (NCAR) National Centers for Environmental Prediction (NCEP), 3) the National Aeronautics and Space Administration (NASA) Land Data Assimilation System (LDAS), 4) the United States Geological Survey (USGS) HYDRO1K dataset, and 5) the PRISM Climate Group PRISM dataset.

This work addresses the challenges of creating metadata for the individual scripts within the VIC data processing workflow shown in Figure 3.1. A significant amount of work by other scientists has gone into creating the software within this workflow, and it is important for the authors of this software to receive credit for their work. It is also important for scientific studies that make use of these lower-level scripts to properly document the specific sequence of software used to perform their analysis. One of the benefits of scientific workflow software (Gil et al., 2007) is capturing the provenance of data processing tasks that support scientific modeling. While workflow software can help to better capture the provenance, it is still important to have sufficient metadata for each step within the workflow. Workflow software alone does not provide this metadata. Instead, the metadata must be populated by scientists and the OntoSoft Ontology can be used to structure this metadata. The methodology section illustrates this process by focusing on the metadata population process for one script within the workflow as an example.

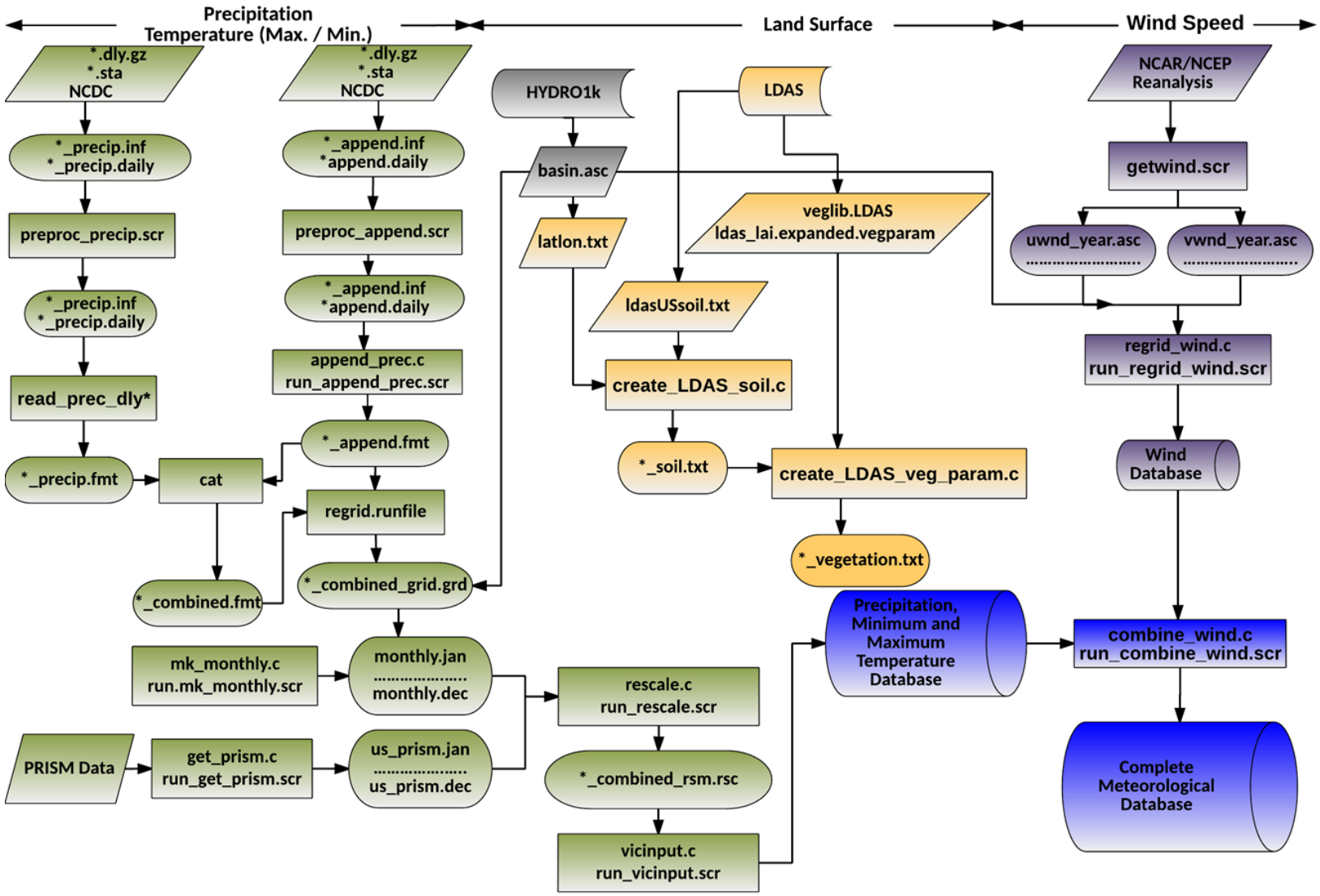



**Figure 3.1** Data pre-processing workflow for the VIC hydrologic model (adapted from Billah et al, 2016).

### *3.2.2 OntoSoft*

OntoSoft consists of an ontology to describe metadata for scientific software (Gil et al., 2015) and the OntoSoft Portal that serves as a user interface to manage that metadata (Gil et al., 2016b). The premise behind OntoSoft's development is that scientific software captures important knowledge and this knowledge should be transparent and shared widely. OntoSoft's ontology and portal support scientists in capturing the important knowledge encapsulated within scientific software. The OntoSoft Portal simplifies the metadata collection process by asking scientists a series of questions. These questions map to specific properties within the ontology. A property defines a relationship (e.g., authorship) between a subject (e.g., the software in question) and an object (e.g. an author). OntoSoft applies the word "software" broadly to include scripts as well as more complex software such as modeling software.

There are 46 properties in the OntoSoft Ontology, equally divided between required and optional properties. These properties are organized into six categories, shown in Figure 3.2. Each category has one or more classes for organizing metadata properties. The six OntoSoft categories are: 1) Identify, 2) Understand, 3) Update, 4) Do Research, 5) Execute and 6) Get Support. The *Identify* category provides a unique description for the software. The *Understand* category describes the metadata needed to increase the trust and domain knowledge about the software. The *Update* category has the metadata to track versioning for the software and how the software is being maintained and developed. The *Do Research* category has the metadata for the input and output data required by the software, relations to other software that can be used with this software, and the software citation. The *Execute* category has the metadata related to how to access, install, and run the software. The *Get Support* category has the contact information for the software developer.


**Figure 3.2** High-level overview of the OntoSoft Ontology (adapted from Gil et al., 2015).

## 3.3 Methodology

The first goal of this study is to extract metadata from various sources in order to create a metadata description for a VIC pre-processing workflow. We consider each step in the workflow to be a unique piece of software with its own metadata description. The second goal of this study is to populate the metadata for each step in the workflow using the OntoSoft Ontology. Five

sources were used for metadata extraction: 1) the source code prior experience running the software, 2) VIC's official website, 3) the software publication in Zenodo, 4) the VIC documentation, and 5) the VIC user discussion wiki. We did not include publications as a metadata source because, after a search of the literature, we only found one publication that discussed VIC pre-processing workflow in any detail, and this study did not include any new metadata beyond what we found in the other five sources. We used only online, publically available resources to populate the ontology and did not contact the software developers. The developers likely could have provided additional metadata for this software, however, a motivation of this research is to better understand what metadata was captured and recorded for this legacy software in online, publically available sources. Once the metadata is extracted, it is then used to populate the ontology through the OntoSoft Portal. The completed documentation includes who authored individual components of the workflow, what the goal of each component was, where each component is published, and other important attributes of the software within a formal, machine-readable form.

### *3.3.1 Using the OntoSoft Portal for Metadata Management*

The OntoSoft Portal was used to insert metadata extracted the from five sources listed above into the OntoSoft Ontology. The OntoSoft Portal presents questions about the software to the scientist, and these questions are mapped to metadata properties in the OntoSoft Ontology. For example, through the OntoSoft Portal, the user is asked "What is the software called?" and the answer to this question is placed as the value for the "has name" property. Table 3.1 shows all the OntoSoft questions as they appear to the scientist on the OntoSoft Portal, along with the property each answer is mapped to. The table also shows the six categories within the OntoSoft Ontology, the classes for each property, and whether the property is required or optional.

### *3.3.2 Example of Metadata Extracted from Source Code*

As an example, the metadata extraction procedure is illustrated for one metadata source (source code and prior experience) and for one software component within the workflow (read_prec_dly). Figure 3.3 shows a screenshot of how the metadata is encapsulated within the software's source code. Metadata extracted from this source code is shown in Table 3.2 and includes the name, programming language, author, and description. The description is interesting because it includes additional metadata information about input and output for the software, as well as workflow composition metadata in terms of upstream and downstream software. From prior experience using the software, metadata including the input and output data file names, operating system software dependencies and other relevant metadata was determined and are listed in Table 3.3.

Once the metadata is extracted, the next step is to map between the extracted metadata and the OntoSoft Ontology. From this one source it is possible to populate 12 of the 46 properties within the OntoSoft Ontology as shown in Figure 3.4. The OntoSoft Portal played an important role in populating the ontology for the software. Figure 3.5, provides an example of how the captured metadata from two different sources, the "source code" source discussed earlier and the "software publication website (Zenodo)" source, were mapped to questions presented through the OntoSoft Portal. The programer names, included as a comment within the source code, were set as the software's creators. The name for the software was assumed to be the file name in this case. The description from the source code was used as the short description of the software. Zenodo (https://zenodo.org/record/22307#.WWjbAYjythE), which hosts this software as a part of the larger VIC source code repository, provides a DOI for the source code. This DOI was used as the

software's unique identifier. The VIC model official website URL is used as the project website for the software.

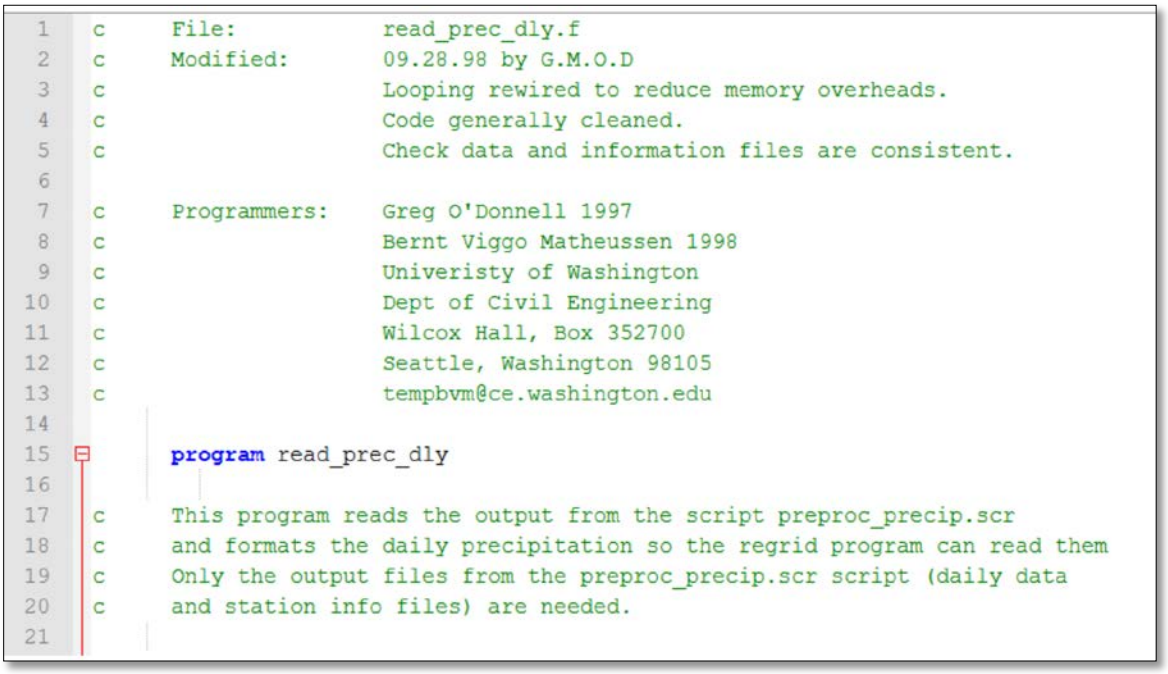
Using additional sources allows for populating the other properties within the OntoSoft Ontology. This procedure was repeated for all metadata sources and all software components to determine the percentage of both the required and optional metadata properties that could be populated from just these publically available sources. As evident in this example, there is a level of interpretation required to perform this mapping. A discussion of the level of confidence in the mapping is reported in the Results and Discussion section along with the results of the metadata extraction process.

**Table 3.1** OntoSoft Portal question and the associated metadata properties within the OntoSoft

| OntoSoft Portal Question | Metadata Properties | Required and Optional Metadata | Class | OntoSoft Metadata Category |
|---|---|---|---|---|
| What is the software called? | has name | Required | Locate | Identify |
| What is a short description for this software? | has short description | | | |
| What are general categories (keywords, labels) for this software? | has software category | | | |
| Is there a project website for the software? | has project web site | | | |
| What is the DOI or any other unique identifier for this software (or software version)? | has unique ID | Optional | | |
| Who created this software? (e.g., Project, Organization, Person, Initiative, etc.) | has creator | Required | Trust | Understand |
| Are there any additional contributors of note for this software? | has major contributor | | | |
| What useful features of this software are worth highlighting? | has salient qualities | | | |
| Who is the publisher of this software if not the author? | has publisher | Optional | | |
| How can a user get support for the software? (e.g., Report bugs, request features and extensions, etc.) | commitment of support | | | |
| Has the software been adopted in a project, organization or by a person? | has adopters | | | |
| Is there any information about uses of this software (e.g., papers, research labs, etc.)? | has use information | | | |
| Are there any statistics of its use? | has use statistics | | | |
| Are there any publications where the software is used? | used in publication | | | |
| Is there any benchmark information about the software? | has benchmark information | | | |
| What are the funding sources for this software? | has funding sources | | | |
| What are the ratings for this software? | has ratings | | | |
| What are domain specific keywords for this software? (e.g., hydrology, climate) | has domain keywords | Required | Relate | |
| Is there any other similar software that you know of? | similar software | Optional | | |
| What are the recommended uses and assumptions for the software? | has uses and assumptions | | | |
| Are there any constraints on use, situations it is not designed for, simplifications? | has use limitation | | | |
| How is the software being developed or maintained? | has active development | Optional | Contribute | Update |
| Are there any on-line resources for accessing the developer community for this software? (e.g., discussion board, wiki, etc.) | has software community | | | |
| What versions does the software have? | has software version | Required | Track | |

**Table 3.1** (continued). OntoSoft Portal question and the associated metadata properties within the OntoSoft

| OntoSoft Portal Question | Metadata Properties | Required and Optional Metadata | Class | OntoSoft Metadata Category |
|---|---|---|---|---|
| What input files does the software require? | has input | Required | Experiment | Do Research |
| What are the input parameters used for this software? | has input parameter | | | |
| What output files does the software produce? | has output | | | |
| Are there any relevant data catalogs that can be used with this software? | has relevant data sources | Optional | | |
| What other software can interoperate with this one? | has interoperable software | Required | Compose | |
| Is this software typically used with other software in a workflow? (e.g., for visualization, preprocessing, post processing, etc.) | has composition description | Optional | | |
| Is there a preferred publication or citation for this software? | has preferred citation | Required | Cite | |
| What is the URL for the code? | has code location | Required | Access | Execute |
| What license is the code released under? | has license | | | |
| Is there a URL for the executable? | has executable location | Optional | | |
| Is there any on-line documentation about the software? | has documentation | Required | Install | |
| What language(s) is the software written in? | has implementation language | | | |
| What Operating Systems can the software run on? | supports operating system | | | |
| How can one install the software? | has installation instructions | | | |
| What other software does the software require to be installed? | has dependency | | | |
| Are there estimates of how long it takes to run this software on average? | has average run time | Optional | | |
| Are there any memory requirements for this software? | requires average memory | | | |
| Are there any other important details about the implementation of this code (e.g., parallelization, special hardware, etc.)? | has other implementation details | | | |
| Is there any test data available for the software? | has test data | Required | Run | |
| Are there any specific instructions for testing the software? | has test instructions | Optional | | |
| What is the e-mail contact for this software? | has email contact | Required | Discuss | Get Support |
| What is the support offered for this software? | has software support | Optional | | |



```
c     File:           read_prec_dly.f
c     Modified:       09.28.98 by G.M.O.D
c                     Looping rewired to reduce memory overheads.
c                     Code generally cleaned.
c                     Check data and information files are consistent.
c
c     Programmers:    Greg O'Donnell 1997
c                     Bernt Viggo Matheussen 1998
c                     Univeristy of Washington
c                     Dept of Civil Engineering
c                     Wilcox Hall, Box 352700
c                     Seattle, Washington 98105
c                     tempbvm@ce.washington.edu

      program read_prec_dly

c     This program reads the output from the script preproc_precip.scr
c     and formats the daily precipitation so the regrid program can read them
c     Only the output files from the preproc_precip.scr script (daily data
c     and station info files) are needed.
```

**Figure 3.3** The header information for the source code of one of the software in the VIC pre-processing workflow. This is a comon approach to include unstructured metadata in scientific software.

**Table 3.2** Metadata extracted from the read_prec_dly.f software's source code

| has name | has creator | has major contributor | has short description | has input | has composition description | has implementation language |
|---|---|---|---|---|---|---|
| read_prec_dly.f | Greg O'Donnell<br>Bernto Matheussen | G.O.M.D | This program reads the output from the script preproc_precip.scr and formats the daily precipitation so the regrid program can read them Only the output files from the preproc_precip.scr script (daily data and station info files) are needed. | daily data<br>Station info files | reads output from preproc-precip.scr Provide input for regrid program | FORTRAN 77 |

**Table 3.3** Metadata captured from experience applying the software.

| has name | used in publication | has input | supports operating system | has output | Has software dependency |
|---|---|---|---|---|---|
| read_prec_dly.f | Billah, M.M., Goodall, J.L., Narayan, U., Lakshmi, V., 2015. Using a Data Grid to Support Regional-Scale Hydrologic Modeling. | Prcp.daily<br>Prcp.inf | Linux | Basin_prcp.fmt | F77 |

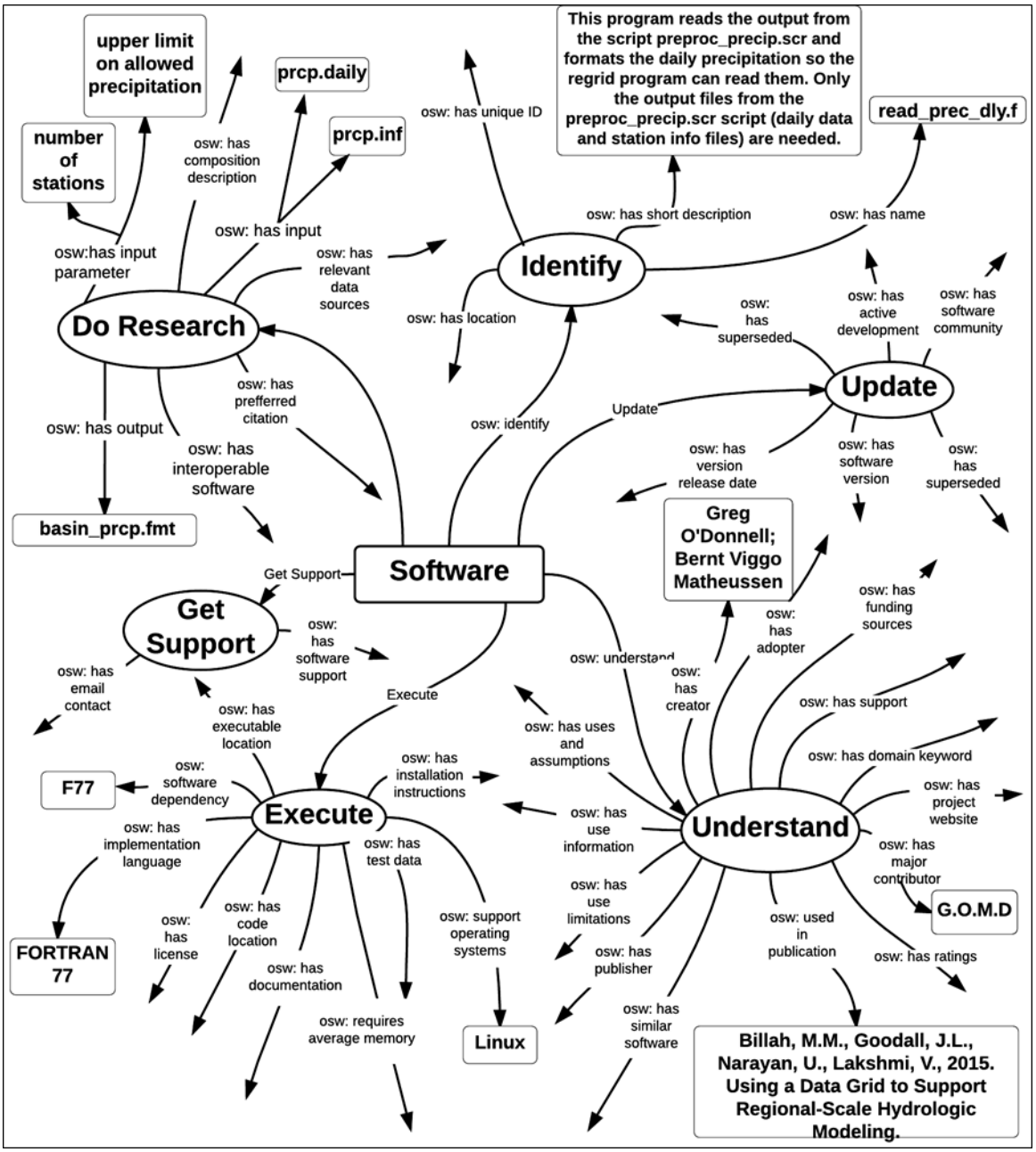



**Figure 3.4** The OntoSoft Ontology for the read_prec_dly software component with properties populated from only one of the five sources: "source code and prior experience." The prefix "osw" denotes to the OntoSoft Vocabulary namespce.

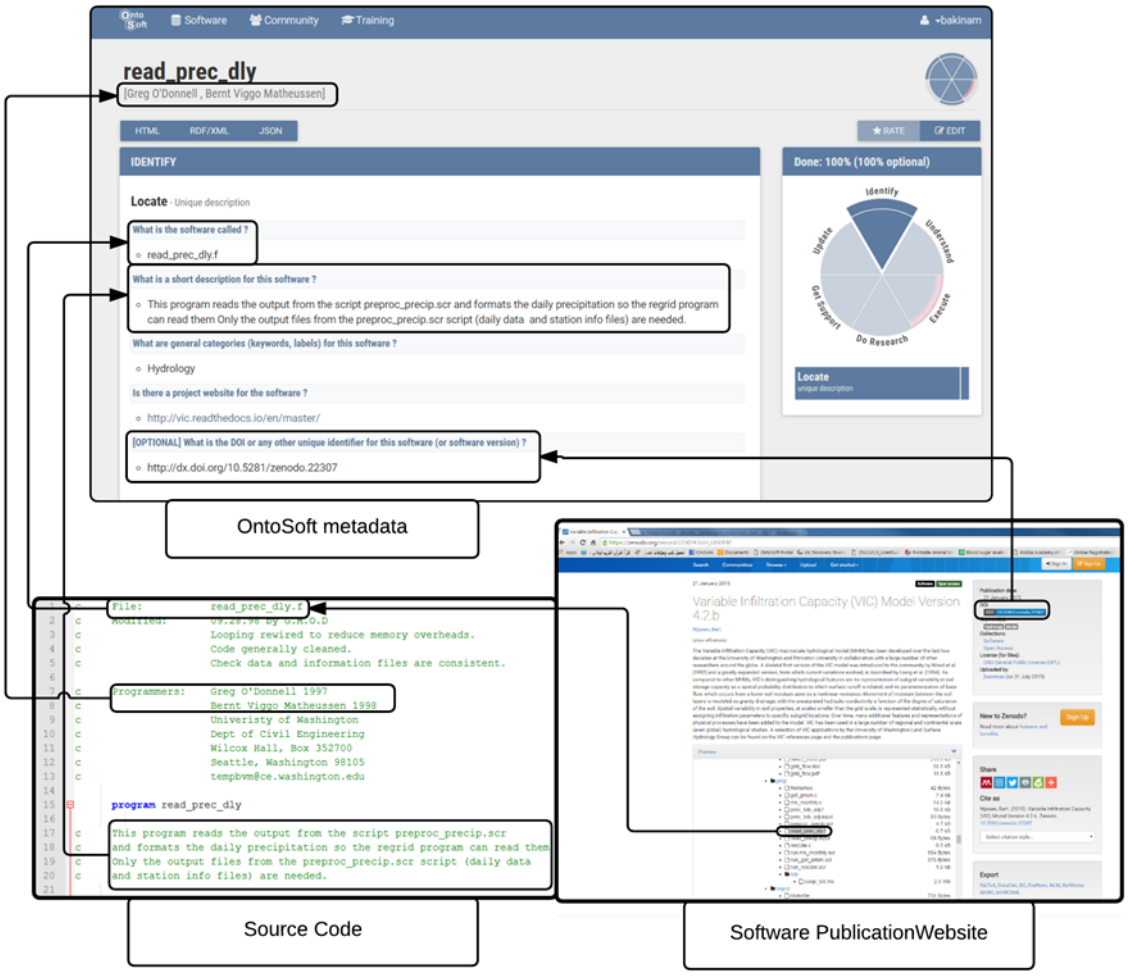



**Figure 3.5** Origin and destination of the captured metadata through the OntoSoft Portal for the identify category.

## 3.4 Results and Discussion

### *3.4.1 Results of the Metadata Extraction*

Figure 3.6 shows the resulting metadata for two of the five OntoSoft categories (Identify and Understand) presented through the OntoSoft Portal for the software component (read_prec_dly) discussed in the Methodology section. The resulting metadata for this software and for the other software components in the VIC data processing workflow are available within

the OntoSoft Portal system. Table 3.4 points to the URLs in the OntoSoft Portal for the 15 software components. The portal provides a user-friendly view of the metadata, but also machine-readable versions of the metadata. The metadata can be viewed using a Resource Description Framework (RDF) eXtensible Markup Language (XML) format or JavaScript Object Notation (JSON) format. These machine-readable formats are built by the system from the data provided by the scientist through the OntoSoft Portal user interface.

**Table 3.4** URL in the OntoSoft Portal for the 15 software within the workflow

| ID | Software | OntoSoft Portal URL |
|---|---|---|
| 1 | preproc_precip | http://ontosoft.org/portal/#browse/Software-11IHopcxMu7x |
| 2 | read_prec_dly | http://ontosoft.org/portal/#browse/Software-3SirBaFht0YN |
| 3 | preproc_append | http://ontosoft.org/portal/#browse/Software-FYMaj4P7bKDb |
| 4 | append_prec | http://ontosoft.org/portal/#browse/Software-hVNbrGnWJ4Zd |
| 5 | run_append_prec | http://ontosoft.org/portal/#browse/Software-GoEvXyadBBVw |
| 6 | regrid | http://www.ontosoft.org/portal/#browse/Software-ZtA35mwlwFmi |
| 7 | mk_monthly | http://ontosoft.org/portal/#browse/Software-DlszQOw6g336 |
| 8 | get_prism | http://ontosoft.org/portal/#browse/Software-vw8DQn2SSnMQ |
| 9 | rescale | http://ontosoft.org/portal/#browse/Software-clQ0WKwjV3Js |
| 10 | vicinput | http://ontosoft.org/portal/#browse/Software-IPXGcujizwTr |
| 11 | create_LDAS_soil | http://ontosoft.org/portal/#browse/Software-AUqV48s3WrgH |
| 12 | create_LDAS_veg_param | http://ontosoft.org/portal/#browse/Software-MZosBxc1Hwl8 |
| 13 | getwind | http://ontosoft.org/portal/#browse/Software-mpNqVzc633VL |
| 14 | regrid_wind | http://www.ontosoft.org/portal/#browse/Software-2QGjMmxS9Du6 |
| 15 | combine_wind | http://ontosoft.org/portal/#browse/Software-ffgkh4iELbOn |

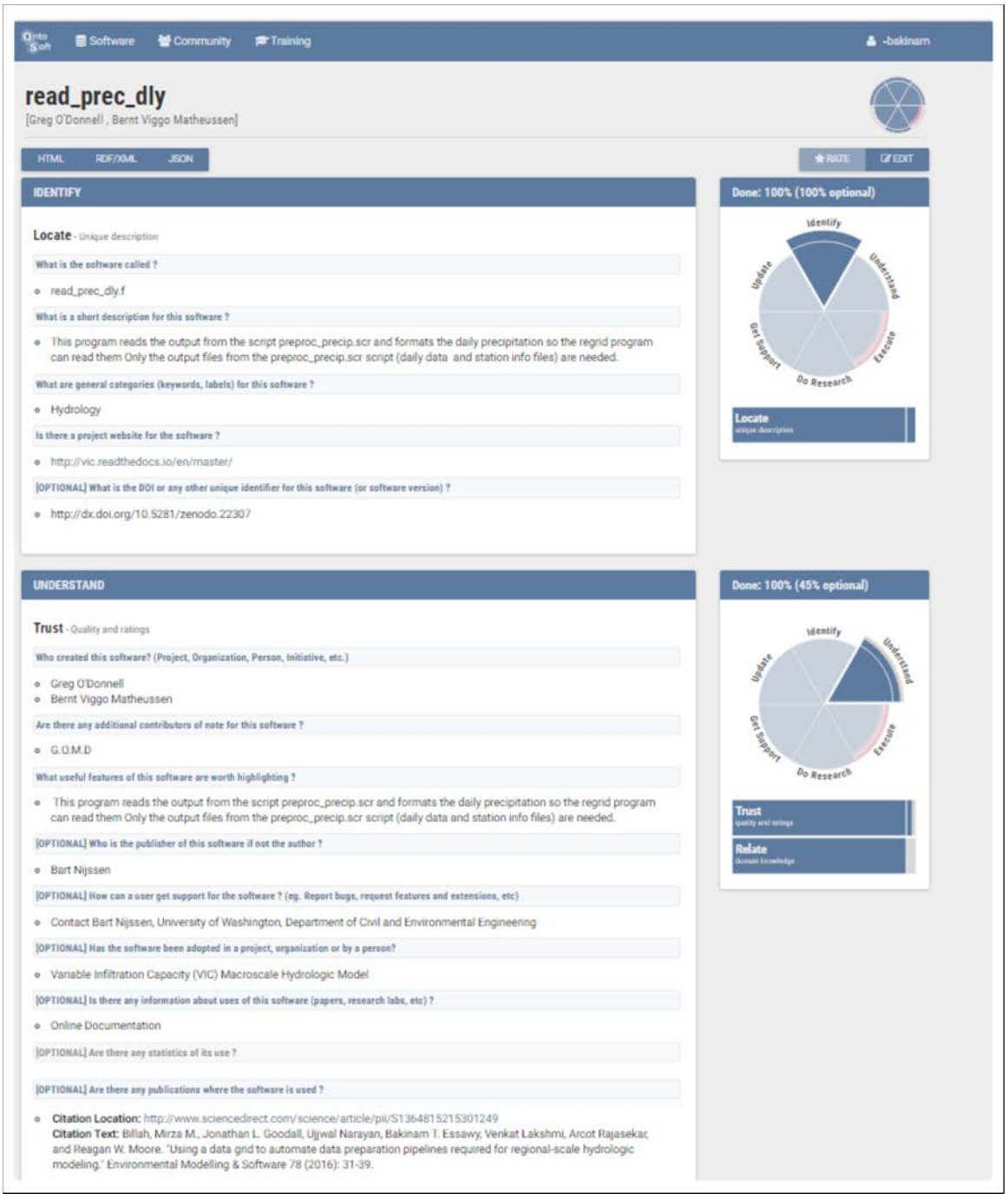



**Figure 3.6** A screenshot for OntoSoft interface showing the captured metadata for the read_prec_dly software within two categories: Identify and a portion of the Trust metadata within the Understand category.

### *3.4.2 Metadata Completeness*

One of the ways the OntoSoft Ontology was evaluated was by recording which OntoSoft properties could be extracted from available online resources for the VIC pre-processing software components. To do this the percentage of metadata completeness for each software within the workflow was calculated and is presented in Figure 3.7 and Table 3.5. The results show that for 13 of the 15 software in the workflow, 74% or more of the metadata mapped to terms in OntoSoft. It seemed that there were consistent practices for including metadata within the software with the exception of two of the software (ID 11 and 12). These two software entries are missing important metadata like author name, function of the software, etc. and only include the source code and few comments within the software itself. These poorly described software entries may have been perceived to play a minor role within the overall software system. This also could have been a result of a difference in practice regarding commenting in the source code for these two software, which were both related to soil and vegetation data preparation.

Table 3.5 also shows that the optional metadata for the Execute category is missing for all software. This category consists of three classes: "Access," "Install," and "Run." These classes depend on the execution of the software with test data like: "has executable location," "has average run time," "requires average memory," and "has test instructions." These properties assume a standalone executable software, but the software analyzed in this study were lower-level software components within a larger software system. It is likely because the software analyzed was at such a fine granular level within the overall model code that such properties are not well documented. We suspect that some of these metadata would likely be available if we took a higher-level view of the software rather than focusing on components of the software system.

**Table 3.5** Percent completeness of OntoSoft required and optional metadata for each OntoSoft category.

| ID | Software | OntoSoft Metadata Categories | | | | | | | | | | | | Average of % complete metadata |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Identify | | Understand | | Execute | | Do Research | | Get Support | | Update | | |
| | | Req | Opt | Req | Opt | Req | Opt | Req | Opt | Req | Opt | Req | Opt | |
| 1 | preproc_precip | 100 | 100 | 100 | 36 | 87 | 0 | 80 | 50 | 100 | 100 | 100 | 100 | 79 |
| 2 | read_prec_dly | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |
| 3 | preproc_append | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 0 | 100 | 100 | 100 | 100 | 78 |
| 4 | append_prec | 100 | 100 | 100 | 45 | 87 | 0 | 80 | 50 | 100 | 100 | 100 | 100 | 80 |
| 5 | run_append_prec | 100 | 100 | 50 | 45 | 87 | 0 | 100 | 0 | 100 | 100 | 100 | 100 | 74 |
| 6 | regrid | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |
| 7 | mk_monthly | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |
| 8 | get_prism | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |
| 9 | rescale | 100 | 100 | 50 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 78 |
| 10 | vicinput | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 78 |
| 11 | create_LDAS_soil | 100 | 0 | 50 | 27 | 87 | 0 | 80 | 50 | 100 | 0 | 0 | 100 | 50 |
| 12 | create_LDAS_veg_param | 100 | 0 | 50 | 27 | 87 | 0 | 60 | 50 | 100 | 0 | 0 | 100 | 48 |
| 13 | getwind | 100 | 100 | 50 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 78 |
| 14 | regrid_wind | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |
| 15 | combine_wind | 100 | 100 | 100 | 45 | 87 | 0 | 100 | 50 | 100 | 100 | 100 | 100 | 82 |

* Req. is required metadata through OntoSoft
* Opt. is for Optional metadata through OntoSoft

Focusing on only the required metadata, the results show that 13 out of 15 software components include 90% or more of the required metadata (Figure 3.7). The optional metadata completeness varied widely among the software between 30% and 66%. Most of the software were downloaded from the Zenodo website except for the software used for soil and vegetation data processing (ID's 11 and 12), which was downloaded from the VIC official website and was not available through Zenodo. Because this soil and vegetation data processing software was not available from Zenodo, it resulted in missing metadata terms associate with software publication (e.g., "has publisher," "has preferred citation"). Also, as discussed earlier, the authors of these software did not include as much metadata within the source code comments compared to other software components. This resulted in the software associated with soil and vegetation data processing lacking metadata compared to the other software components.

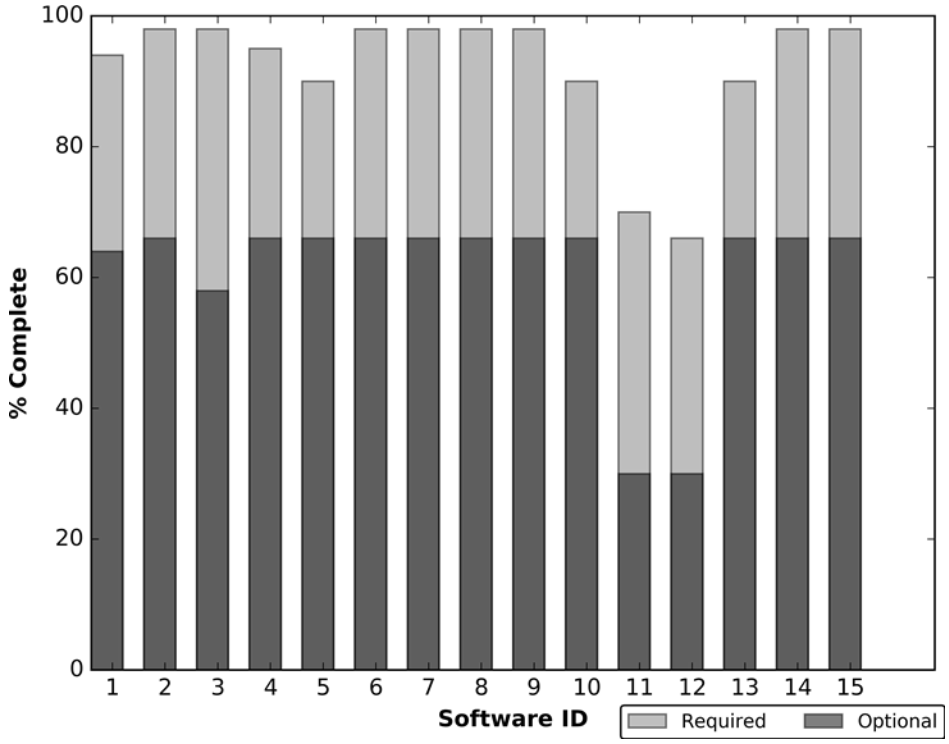



**Figure 3.7** Percent Completeness of OntoSoft required and optional metadata for each software in the VIC pre-processing workflow.

There are common metadata that are missing from all of the software components. Table 3.6 shows the 10 optional and 1 required properties that were missing for all the software. The one missing required property, "has test data," was not identified for any of the software through this research, as discussed earlier. It may be necessary to make this an optional rather than required property for more modular software components. Test data should always be included, even to support unit tests of modular components of a larger software system. However, given that this may not have been a common practice in the past, making this optional metadata to support legacy codes may be appropriate. Of the 10 missing optional properties, all are important but none could be captured for this software based on our analysis of available online resources. Some of the missing optional properties may be difficult to populate for other software as well, because they will be heavily dependent on applications of the software to specific use cases (e.g., "has average run time" and "requires average memory").

**Table 3.6** Common missing metadata across software in the workflow.

| Metadata Properties | Required and Optional Metadata | Class | OntoSoft Metadata Category |
|---|---|---|---|
| has use statistics | Optional | Trust | Understand |
| has benchmark information | | | |
| has funding sources | | | |
| has ratings | | | |
| similar software | Optional | Relate | |
| has uses and assumptions | | | |
| has use limitation | | | |
| has executable location | Optional | Access | Execute |
| has average run time | Optional | Install | |
| requires average memory | | | |
| has test data | Required | Run | |
| has test instructions | Optional | | |

### *3.4.3 Metadata Sources*

Another interesting outcome of the results is a better understanding of the percentage of metadata that comes from each of the five sources used for metadata extraction Figure 3.8. The "source code and prior experience" source provided the most metadata. The VIC documentation provided nearly the same amount of metadata as the software publication in Zenodo provided. Collectively, these three sources supplied 80% of the metadata with the other 20% being supplied by the VIC website and user discussion wiki. The results show how the metadata is distributed across the sources and further argues for the need to centralize metadata for hydrologic modeling software.

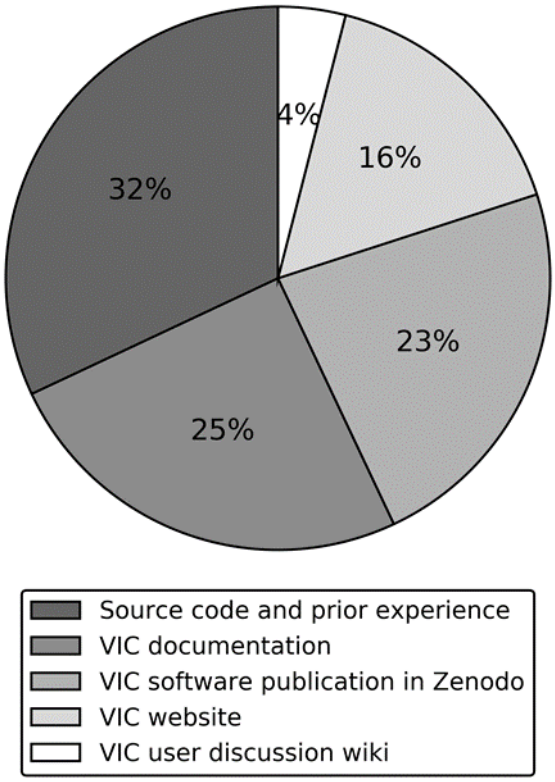



**Figure 3.8** Percentage of extracted metadata coming from each of the five sources

When the metadata source data is broken down by OntoSoft categories, it is clear that some sources play a more major role than others in populating each category's metadata Figure 3.9. For

example, the VIC website was only used to populate metadata in the Update category. The VIC documentation and documentation were used to populate metadata in five of the six categories; no source was used in all six categories. Interestingly, metadata for Identify, Execute, and Do Research categories came from the same three sources: the VIC publication in Zenodo, the VIC documentation, and the source code and prior experience. This result shows how valuable metadata is being captured now, but even when broken into thematic categories, metadata is still widely distributed across sources.

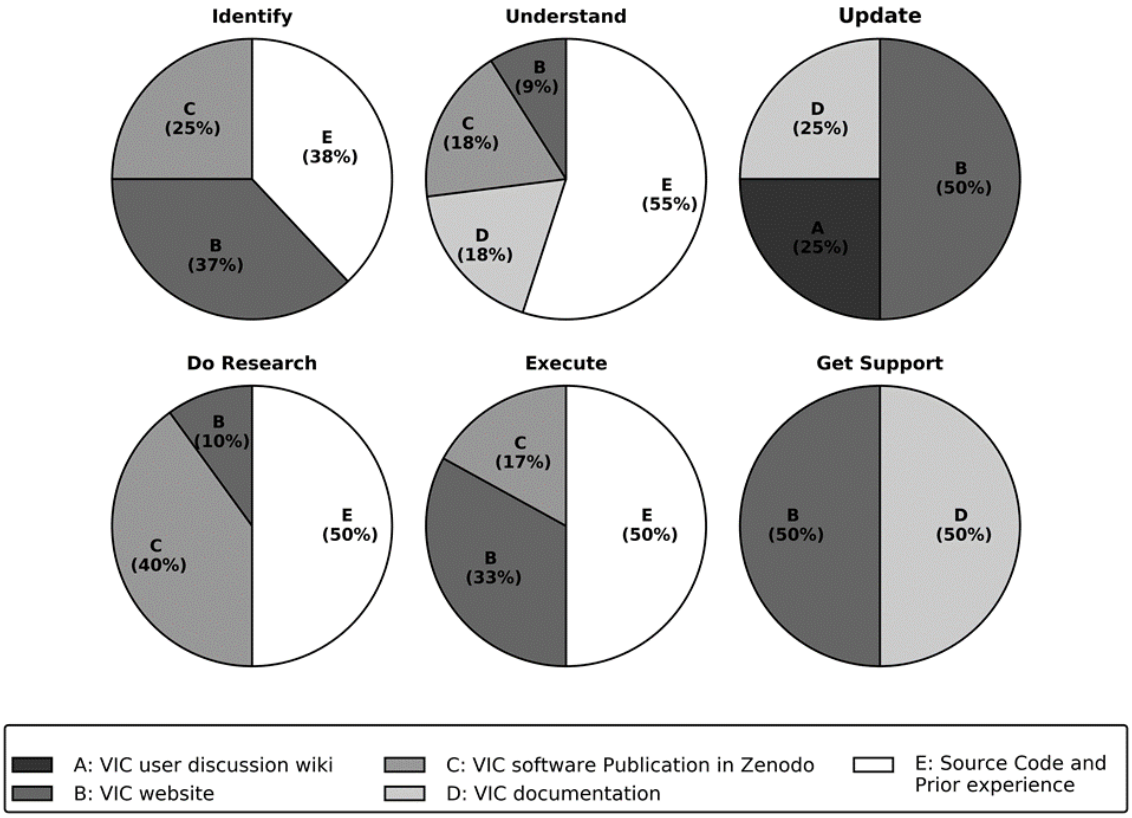



**Figure 3.9** Source for extracted metadata for each OntoSoft Category.

### *3.4.4 Confidence in Metadata Mapping*

Some the mappings for ontology properties are uncertain, meaning it is expected that not all will agree with how extracted metadata was mapped to ontology properties in this study. Table 3.7 shows the level of confidence the authors had for the ontology property mapping completed in this study. Some properties have high confidence, where it is likely others performing this same metadata extraction exercise would arrive at the same result. Other properties were rated as low confidence, meaning it is likely, in the opinion of the authors, that others may populate these fields differently than what was done in this study. In some cases, the low confidence properties for this study may have higher confidence if this procedure was completed for another model software. In other cases, the low confidence properties were the result of ambiguity as to how metadata from available sources should be mapped to these properties. These properties may require further consideration and explanation for use with hydrologic modeling.

**Table 3.7** Level of confidence in metadata properties populated on OntoSoft

| OntoSoft Category | High Confidence | Low Confidence |
|---|---|---|
| *Identify* | has name<br>has project web site<br>has unique ID | has short description<br>has software category |
| *Understand* | has creator<br>has publisher | has major contributor<br>has short description<br>commitment of support<br>has domain keywords<br>has use limitations<br>has use information<br>used in publication<br>has salient qualities |
| *Update* | has software version<br>has active development<br>has software community | has version release date<br>supersedes<br>superseded by |
| *Do Research* | has input<br>has input parameter<br>has output<br>has preferred citation | has relevant data sources<br>has interoperable software<br>has composition description |
| *Execute* | has code location<br>has license<br>has documentation<br>has implementation language<br>has dependency<br>supports operating systems | has installation instructions |
| *Get Support* | has email contact | has software support |

## 3.5 Conclusion

This work evaluates the OntoSoft Ontology and portal for capturing and sharing metadata for legacy hydrologic modeling software. The OntoSoft Ontology is designed to focus on scientists rather than software developers (Gil et al., 2015), so it is important for scientists to evaluate the ontology. This work also supports the idea of sharing software and its associate metadata as an additional goal to complement the now commonly accepted idea of sharing data and its associate metadata. To achieve "reproducible software" (Peng, 2011), not only the software and data need to be shared, but also their associated metadata. Sharing software with metadata encourages future scientists to learn and build from prior work by reducing the time and effort to find and understand this prior work. This study uses a pre-processing workflow for the VIC hydrologic model as a case

study for evaluating the OntoSoft Ontology. Metadata was harvested from five sources: 1) Source code and prior experience, 2) Variable infiltration capacity (VIC) model official website, 3) Software published in website Zenodo, 4) VIC documentation for the software, and 5) VIC user discussion wiki. The large amount of effort and time devoted to capturing metadata from these various sources resulted in an improved description of the complex hydrologic VIC model workflow at a detailed level using the OntoSoft Ontology.

Results of the analysis showed that at least 90% of the required OntoSoft metadata properties could be captured from the online sources for 13 of the 15 software components within the workflow. The metadata was somewhat evenly distributed across four of the five sources. This result suggests that the vast majority of the metadata needed to populate at least the required properties in OntoSoft is recorded now by hydrologic modelers, but the information is distributed across sources and stored in unstructured forms. This study also showed that there are common missing properties across all the software used within the workflow. Out of 46 properties in the OntoSoft Ontology, there were 14 optional properties (< 30%) and one required properties (< 3%) missing for all 15 software. Some of the missing properties (e.g., memory size and run time) depend on a specific application of the software (i.e., to model a given domain for addressing a given research objective), and thus will differ from one application to another. Finally, the results of the study also suggested uncertainty in how to populate some of the metadata properties. Some of these terms, labeled as "low confidence" in Table 3.6, may have had less uncertainty if a different set of software were investigated (e.g., software at less of a fine-grain level than what was used in this study). Other terms may be ambiguous across hydrology models, requiring additional description and guidance.

Some limitations of this study are that (i) while it investigates 15 different software, these are all related to using a single hydrologic model and (ii) the metadata was extracted by one team of hydrologists. Broadening this work to additional geoscience models and having other scientists repeat the metadata extraction process would help to advance the evaluation of OntoSoft for capturing geoscience software metadata. In particular, having other groups of scientists repeat the process would benefit in testing the consistency of the metadata property mapping process. Expanding the effort to other geoscience models would help in improving the evaluation of OntoSoft for representing the metadata necessary for geoscience software more broadly. Despite these limitations, this study contributes both an important and necessary evaluation of OntoSoft as ontology for describing software relevant to hydrologic modeling. It also improves understanding of what metadata is being captured now in available online resources for hydrologic modeling software.

Finally, there are many possible future research goals that could be undertaken to advance the research presented here. 1) OntoSoft could be expanded to better track where metadata recorded within the ontology was obtained. 2) The extraction process, which is now manual and very tedious, could be more automated through text mining approaches, although from this experience we believe manual intervention will continue to be necessary at some level. 3) For the low confidence metadata, a mechanism for crowdsourcing the metadata collection and review (potentially through a user-supplied rating system) would be a helpful feature for gaining confidence in potentially ambiguous metadata. 4) Experiments, where a group of scientists repeat the same procedure outlined in this study for gathering metadata on the VIC pre-processing workflow and entering it through the OntoSoft Portal, would be a potentially useful way to compare the completeness, confidence, and accuracy of metadata generation across scientists.

Lastly, an underlying premise of this study is that having metadata for software, including for software at a fine-grain level, is useful for increasing transparency and reproducibility in science. Future work could test this assumption by surveying VIC users to better evaluate how metadata presented through the OntoSoft Portal increases their understanding of the VIC software, and how it influences their use and communication of the software with other researchers going forward.

## 3.6 References


Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environ. Model. Softw. 78, 31–39.

Cassey, P., Blackburn, T.M., 2006. Reproducibility and Repeatability in Ecology. Bioscience 56, 958–959.

David, C.H., Gil, Y., Duffy, C.J., Peckham, S.D., Venayagamoorthy, S.K., 2016. An introduction to the special issue on geoscience papers of the future. Earth Sp. Sci. doi:10.1002/2016EA000201.Received

De Roure, D., Goble, C., Stevens, R., 2009. The design and realisation of the Virtual Research Environment for social sharing of workflows. Futur. Gener. Comput. Syst. 25, 561–567. doi:10.1016/j.future.2008.06.010

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. doi:10.1002/2015EA000139

Fulweiler, R.W., Emery, H.E., Maguire., T.J., 2016. A workflow for reproducing mean benthic gas fluxes. Earth Sp. Sci. 3, 318–325. doi:10.1002/2015EA000158

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016a. Towards the Geoscience Paper of the Future : Best Practices for Documenting and Sharing Research from Data to Software to Provenance. Earth Sp. Sci. 1–75. doi:10.1002/2015EA000136

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J., 2007. Examining the challenges of scientific workflows. Ieee Comput. 40, 26–34. doi:10.1109/MC.2007.421

Gil, Y., Garijo, D., Mishra, S., Ratnakar, V., 2016b. OntoSoft : A Distributed Semantic Registry for Scientific Software. Proc. Twelfth IEEE Conf. eScience, Balt. MD.

Gil, Y., Ratnakar, V., Ca, R., Garijo, D., 2015. OntoSoft : Capturing Scientific Software Metadata, in: Eighth ACM International Conference on Knowledge Capture, Palisades, NY, 2015.

Heidorn, P.B., 2008. Shedding Light on the Dark Data in the Long Tail of Science. Libr. Trends 57, 280–299. doi:10.1353/lib.0.0036

Higgins, S., 2007. Using Metadata Standards [WWW Document]. Digit. Curation Cent. URL http://www.dcc.ac.uk/resources/briefing-papers/standards-watch-papers/using-metadata-standards#2 (accessed 5.10.16).

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M.J., Tarboton, D.G., 2015. Hydroshare: Sharing diverse environmental data types and models as social objects with application to the hydrology domain. JAWRA J. Am. Water Resour. Assoc. 52. doi: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. 50. doi:10.1002/2016WR019285

JB, G., PJ, G., SJ., W., 2007. OpenMI: Open modelling interface. J. Hydroinformatics 9, 175–191.

Liang, X., Lettenmaier, D.P., Wood, E.F., 1996. One-dimensional statistical dynamic representation of subgrid spatial variability of precipitation in the two-layer variable infiltration capacity model. J. Geophys. Res. Atmos. 101(D16), 21403–21422.

Lud, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., 2006. Scientific workflowmanagement and the Kepler system. Concurr. Comput. Pract. Exp. 18, 1039–1065. doi:10.1002/cpe.994

Mcdougal, R.A., Bulanova, A.S., Lytton, W.W., 2016. Reproducibility in Computational Neuroscience Models and Simulations. IEEE Trans. Biomed. Eng. 63, 2021–2035.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the 7th International Congress on Environmental Modelling and Software, DP Ames, NWT QuinnMorsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the. pp. 978–988.

NISO, N., 2004. Understanding Metadata. Natl. Inf. Stand. Organ. 20. doi:10.1017/S0003055403000534

Peckham, S.D., Goodall, J.L., 2013. Computers & Geosciences Driving plug-and-play models with data from web services : A demonstration of interoperability between CSDMS and CUAHSI-HIS. Comput. Geosci. 53, 154–161. doi:10.1016/j.cageo.2012.04.019

Peckham, S.D., Hutton, E.W.H., Norris, B., 2013. A component-based approach to integrated modeling in the geosciences : The design of CSDMS. Comput. Geosci. 53, 3–12. doi:10.1016/j.cageo.2012.04.002

Peng, R.D., 2011. Reproducible research in computational science. Science. 334, 1226–1227.

Pope, A., 2016. Reproducibly estimating and evaluating supraglacial lake depth with Landsat 8 and other multispectral sensors. Earth Sp. Sci. 3, 176–188. doi:10.1002/2015EA000125

Ratnakar, V., Gil, Y., 2015. OntoSoft [WWW Document]. URL http://ontosoft.org/ontology/software/ (accessed 1.11.16).

Roure, D. De, Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Hull, D., Michaelides, D., Newman, D., Procter, R., Lin, Y., 2010. Towards open science : the myExperiment approach. Concurr. Comput. Pract. Exp. 22, 2335–2353. doi:10.1002/cpe

Scholten, Huub, Waveren, R.H. Van, Groot, S., Geer, F.C. Van, Wösten, J.H.M., Koeze, R.D., Noort., J.J., 2000. Good Modelling Practice in water management, in: In Paper Presented on Hydroinformatics. pp. 23–27.

Singh, V.P., Asce, F., Woolhiser, D.A., Asce, M., 2002. Mathematical Modeling of Watershed Hydrology. J. Hydrol. Eng. 7, 270–292.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Band, L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D., 2014. HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing. Int. Environ. Model. Softw. Soc. 7th Int. Congr. Environ. Model. Software, San Diego, California, USA. www. iemss. org/society/index/php/iemss-2014-proceedings. doi:10.13140/2.1.4431.6801

Yu, X., Duffy, C.J., Rousseau, A.N., Bhatt, G., Álvarez, Á.P., Charron, D., 2016. Open science in practice: Learning integrated modeling of coupled surface-subsurface flow processes from scratch. Earth Sp. Sci. 3, 190–206. doi:10.1002/2015EA000155

# Chapter 4: Integrating Scientific Cyberinfrastructure to Improve Reproducibility in Computational Hydrology: Example Using HydroShare and GeoTrust[3]

## 4.1 Introduction

The challenge of creating more open and reusable code, data, and formal workflows that allow others to verify published findings is gaining attention in the scientific community (Borgman, 2012; Cédric H. David et al., 2016; Gorgolewski and Poldrack, 2016; Peng, 2011; Qin et al., 2016). Recent papers have argued the need for and have proposed approaches to improve reproducibility both broadly within geosciences generally and the hydrologic sciences specifically (Cédric H. David et al., 2016; Essawy et al., 2016; Gil et al., 2016a; Hutton et al., 2016). Here we consider reproducibility to be the ability to document and share digital resources used to complete an analysis including (1) raw initial datasets, (2) data preprocessing scripts used to clean and organize the data, (3) model inputs, (4) model results, and (5) the specific model code along with all of its dependencies (Figure4.1). These data and software are often integrated into workflows that allow scientists to re-run an analysis from raw initial datasets and obtain the same model results.

There are different requirements for reproducibility depending on the nature of the research. For example, empirical reproducibility requires capturing descriptive information about protocols and methods for laboratory-based scientific experiments. Computational reproducibility, the subject of this study, requires descriptive information about the software and workflow details of model-based research (Todden, 2013). Any computational reproducibility solution must be

[3]This Chapter is in preparation for submission to a peer reviewed journal. The tentative title, authors, and journal for the submission follow. Bakinam T. Essawy, Jonathan L. Goodall, Wesley Zell, Daniel Voce, Mohamed M. Morsy, Jeffrey Saddler, and Tanu Malik. Integrating Scientific Cyberinfrastructure to Improve Reproducibility in Computational Hydrology: Example Using HydroShare and GeoTrust. In preparation for submission to Environmental Modelling & Software.

general and able to address this heterogeneous landscape of tools and approaches used within the target scientific community. In hydrology, scientists use a large variety of computational models and many of these computational models have decades of development effort behind them (Singh et al., 2002). Computational modeling requires a significant amount of effort and time to prepare model input, and to calibrate and validate model parameters. These aspects of hydrology make computational reproducibility particularly challenging.

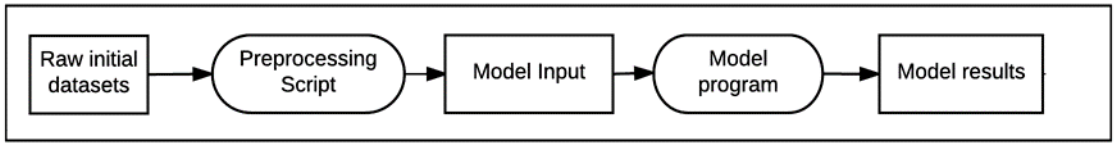



**Figure 4.1** The typical conceptual workflow that needs to be repeated for computational reproducibility.

HydroShare and GeoTrust are two new cyberinfrastructures under active development that can be used to improve reproducibility in computational hydrology. HydroShare is a web-based system for sharing hydrologic data and model resources including detailed, hydrologic-specific resource metadata (Tarboton et al., 2014a, 2014b). GeoTrust provides tools for scientists to efficiently reproduce and share geoscience applications by building "sciunits," which are efficient, lightweight, self-contained digital packages of computational workflows that can be repeated or reproduced in different environments regardless of deployment issues (Hai et al., 2017). However, we believe that neither achieves computational reproducibility as we defined previously in isolation. This paper discusses how Hydroshare and GeoTrust can be harnessed to provide the theoretical notion of computational reproducibility as defined earlier in the domain of hydrology. Hydroshare allows scientists to share datasets (e.g., raw initials datasets, data preprocessing scripts used to clean and organize the data, model software, and model results) in an open and transparent way. It allows for actions on data through apps that view or analyze data stored in HydroShare.

GeoTrust allows scientists to document their computational workflows as sciunit containers. *Sciunit-CLI* (https://bitbucket.org/geotrust/sciunit-cli), a tool, software used to create a sciunit, tracks and outputs the provenance of the computational workflows. The tool also keeps track of software dependencies and creates a container for workflows using Docker. Docker containers allow scientist to wrap a piece of software in a complete filesystem that contains everything needed to run, including code, runtime, system tools, and system libraries (Owsiak et al., 2017). Typically documenting all code, data, and environment dependencies can be burdensome for a scientist, and *Sciunit-CLI* automates this process, taking execution time that is imperceptibly more than the execution time of the workflow. The sciunit also encapsulates retrospective provenance of the workflow execution, which can be used for re-running containers (Pham et al., 2014). Finally, the container can be saved as a Docker or Vagrant container which improves reproducibility in different environments.

The aim of this research is to present a solution for achieving a higher level of reproducibility research using GeoTrust's *Sciunit-CLI* and HydroShare. The solution described in this study can be used to assists scientists to more easily repeat, reproduce and verify a computational experiment (Malik, 2017). This higher level of reproducibility is not limited to being open and simply sharing but also being portable in different environments and repeating analyses with different datasets. It is not possible to share code, data, and environment. when using only HydroShare or GeoTrust is used in isolation. GeoTrust does not provide a community of users who can verify analyses and a variety of datasets that are required for verification that can be achieved by using HydroShare. Hydroshare simply assumes that reproducibility requirements are satisfied if code and data is shared, but reproducibility also involves sharing the environment, which scientists have a hard time documenting, and then repeating with different datasets. This

paper presents a design, prototype implementation, and example application of the approach using MODFLOW-NWT model as a use case. MODFLOW is the U.S. Geological Survey's three-dimensional (3D) finite-difference groundwater model and is commonly used by hydrogeologists to simulate groundwater flow; "NWT" signifies a version of MODFLOW that uses the Newton-Raphson method to solve the system of equations that result from a numerical simulation of the general equation for groundwater flow. To achieve better reproducibility for MODFLOW-NWT, we first use the *Sciunit-CLI* to create a sciunit for the pre-processing workflow used to prepare the input data for a MODFLOW-NWT model for a study area in the James River in Virginia. Then HydroShare is used to share key resources associated with this modeling application to foster reproducibility. Finally, HydroShare is used to initiate the execution of the GeoTrust containers on a cloud-computing environment.

The remainder of the study is organized as follows. First, additional background on the HydroShare, GeoTrust, and MODFLOW-NWT software projects is provided. This background section is meant to orient readers on key aspects of these projects. Next the design and implementation of the HydroShare and GeoTrust integration approach is presented and demonstrated using the MODFLOW-NWT model as an example application. Finally, a discussion and conclusions section summarizes key aspects of the approach and outlines opportunities for future research to advance on known limitations of the approach.

## 4.2 Background

### *4.2.1 HydroShare*

HydroShare (https://www.hydroshare.org) is a web-based system designed to enable hydrologic scientists to easily share, collaborate around, and publish all types of scientific data and models (Tarboton et al., 2014a, 2014b). Digital content is stored in HydroShare as resources and

every resource has a resource type (Horsburgh et al., 2015). The "generic" resource type supports the Dublin Core metadata standard and more specific resource types expand on this metadata for well-defined resource types. For example, "Operating System" is one of the extended metadata terms for the 'Model Program' resource type, which is used for sharing a computational model program in HydroShare (Morsy et al., 2017).

HydroShare provides a web representational state transfer (REST) application program interface (API) that allows third-party applications to interact with HydroShare resources. Developers can also create web-apps to integrate functionality into HydroShare. Web-app developers can catalogue their apps in HydroShare via the 'Web-app' resource type. When a developer creates a web-app resource in HydroShare, the developer specifies which resource types are relevant to the app and the URL that will be called when the app is executed from the landing page of the resource that the app is acting on. After a developer adds his or her web-app as a resource in HydroShare, HydroShare users can execute that app through HydroShare's web interface to act on relevant resources that they have access to.

Although there are several different resource types supported by HydroShare, two of the main resource types relevant to this study deal with computational models. HydroShare divides computational models into two separate but linked resource types: a) the model program and b) the model instance (Morsy et al., 2017). The model program is the software for executing the model and the model instance is the input files required for executing the model and, optionally, the output files after a model instance has been executed by a model program (Horsburgh et al., 2015; Morsy et al., 2014b). A model instance resource can be linked to a model program resource with the model instance extended metadata term "ExecutedBy." Using the "ExecutedBy" term, a user can specify which model program is meant to execute the input files contained within the

model instance resource (Morsy et al., 2017). Additional HydroShare resource types used in this study include the composite resource type, which allows uploading metadata files at both file and resource level, the collections resource type, which stores any number of resources as a single, aggregate resource, and the web-app resource type, which was described earlier.

### *4.2.2 GeoTrust*

The GeoTrust project aims to create cyberinfrastructure that assists scientists in creating and maintaining collections of sciunits that pertain to a specific research project (Malik et al., 2017). Sciunits are computational research objects, that monitor the reference execution of an ad hoc workflow to track and bundle all code, data, and environment dependencies into a light-weight, self-contained container that can be repeated in other environments (Malik, 2017). A sciunit advances the concept of a research object, which is an aggregation of digital artifacts such as code, data, scripts, and temporary experiment results that together with the paper provide an authoritative and far more complete record of a piece of research (Bechhofer et al., 2013). Further, users can attach additional annotations to describe containers. Each container also incorporates associated provenance, and users can use the included provenance to create smaller containers or repurposed containers (i.e. they can create arbitrarily new containers). These containers enable exact or partial repeatability of the sciunit. The objective of GeoTrust project is to provide easy to use tools to create, store, and manage sciunits.

Sciunits are Docker containers used to improve the reproducibility of a computational analysis by providing a single container for handling the various model-related data items and software components used during a computational analysis. Data and software elements may include input files, parameter files, pre- or post-processing scripts, the model executable, any associated libraries, and all output files produced by the model and scripts. The objective of

GeoTrust is to develop a way to define, share, and access the needed metadata collections for each file used in the computational analysis. This approach aims to enhance sharing, reusing, and reproducibility of models not just in hydrology but more broadly within the scientific community.

The GeoTrust project provides a tool called *Sciunit-CLI* that automatically creates the sciunit Docker containers for scientific applications and runs in any Linux environment. *Sciunit-CLI* is a Python/C command-line interface. Using *Sciunit-CLI* to create a sciunit is a way for scientists to document their workflow, share it with others, and publish it on publicly accessible repositories. When a sciunit is created, *Sciunit-CLI* ensures that the package contains all the dependencies required for the workflow, the sciunit can be rerun, and the outputs reproduced, using any other deployment configuration that also has *Sciunit-CLI* installed. Because it contains all the required dependencies. When *Sciunit-CLI* creates a sciunit, it includes three types of metadata: annotation metadata (populated by the user) and provenance and version metadata (generated automatically by *Sciunit-CLI*). This process is further described with an example application later in the study.

Figure 4.2 shows a sample user interaction with this client. The user instantiates a namespaced sciunit titled *myro,* and can associate files and annotations with the sciunit using *CLI* commands shown in (in italics) in Figure 2. To create a container within the sciunit, bundling an application's digital artifacts, the user runs the application with the *package* command. The user application can be written in any combination of programming languages e.g. C, C++, Fortran, Shell, Java, R, Python, Julia, etc. In our example, the application consists of the data pre-processing scripts written in R and Python. Packaging an application also incorporates provenance information. Many such containers can be created within the same sciunit. The client works in a

Git-like fashion in that the *myro* sciunit is stored only locally unless it is explicitly shared with a remote repository. This method of operation allows distributed collaborators to work offline on the same sciunit. When a user is ready to share, she can publish a container to a remote sciunit using the *publish* command, which instructs the client to upload the container to a Web-based repository. The repository reads the container's contents, stores the container's digital artifacts in the appropriate remote sciunit, and associates the container with an appropriate cloud execution server on which it can potentially re-execute.

A container within the sciunit can be re-run directly from the client, either locally on the local machine with the *repeat* command, or remotely on a remote execution server with the *repeat remote* command, as shown in Figure 4.2. In the remote case, the target container is downloaded from a Web-based repository to a remote execution server, and, if the container is compatible with the execution server's architecture, the execution server runs it and sends the results back to the user. Both local and remote executions may optionally be repeated as partial executions. Finally, the user can modify a container by downloading it, modifying its code or data and running it locally, and then uploading the modified container, at which point a new version of the container will be stored in the Web-based repository.

```
1. > geounit start FIE
2. > annotate FIE author:Tanu Malik
3. > track FIE.sh 1 /default/data
4. > package FIE.sh 1 /default/data
package_hash = 04er667
5. > publish 04er667
enter abstract: FIE package
title: Food Inspection Evaluation
keywords (split with comma): food, FIE
do you want to make this resource
public (y/n):y
resource was created
6. > repeat 04er667
7. > repeat remote 04er667
8. > stop
```

**Figure 4.2.** User interaction with sciunit client.

### *4.2.3 MODFLOW-NWT*

MODFLOW is the U.S. Geological Survey's (USGS) three-dimensional (3D) finite-difference groundwater model. The USGS has released multiple versions of MODFLOW. MODFLOW-2005 is the most widely used and most thoroughly tested version of MODFLOW. MODFLOW-NWT is an advanced MODFLOW version that includes specialized MODFLOW variants and uses a Newton-Raphson formulation to improve the solution of unconfined groundwater-flow problems. MODFLOW-NWT is a standalone program that is intended for solving problems involving drying and rewetting nonlinearities of the unconfined groundwater-flow equation (Niswonger et al., 2011). MODFLOW-NWT packages have nearly the same format as the standard MODFLOW-2005, with a few exceptions (e.g., the list of possible Newton-Raphson solver input variables is more extensive than most MODFLOW solvers).

This study leverages FloPy, a Python library that allows users to perform pre-processing routines to create new MODFLOW models from raw datasets, to run MODFLOW models using a variety of different model versions, and to post-process model output files (Bakker et al., 2016). Using scripting for these steps as opposed to a Graphic User Interface (GUI), which is commonly

used for geoscience models, makes the data processing steps more transparent and reproducible. By combining FloPy with GeoTrust and HydroShare, the workflow used to create, execute, and analyze the output of a MODFLOW model (e.g., the steps shown in Figure 4.1) can be stored within a reproducible container with descriptive metadata in HydroShare, as described in the following sections.

## 4.3 System Design and Implementation

Figure 4.3 shows the activity diagram with a high-level view of the system designed for integrating GeoTrust and HydroShare. First, a user logs into a machine with *Sciunit-CLI* installed and configured. The user then starts *Sciunit-CLI* at the terminal. While *Sciunit-CLI* is running, the user initiates a workflow, often a shell script that executes a series of Python scripts for automating data processing steps associated with a modeling use case. *Sciunit-CLI* creates a sciunit (Docker container) that includes all software and data dependencies for the executed workflow. The scientist can then use *Sciunit-CLI* to automatically share the sciunit through HydroShare with basic resource metadata.

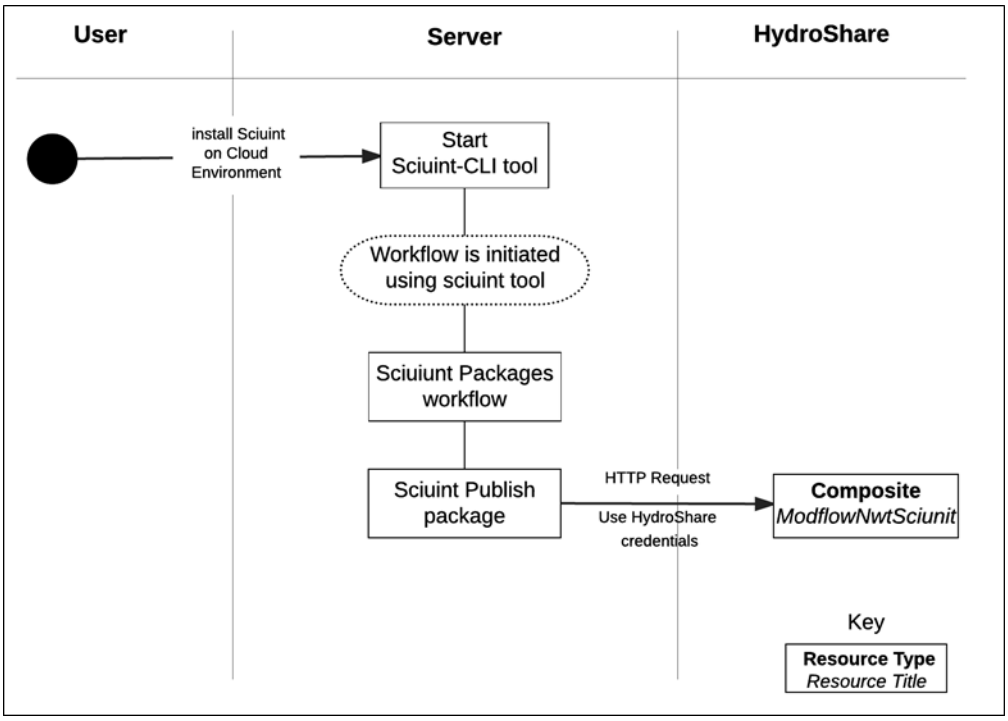



**Figure 4.3** Activity diagram integration showing the creation of a sciunit using GeoTrust and publishing on HydroShare.

In order to implement this design, *Sciunit-CLI* needed to be extended to support sharing of sciunits through HydroShare. This functionality was implemented using HydroShare's representational state transfer (REST) application program interface (API). To store a sciunit on HydroShare through *Sciunit-CLI*. The user must provide valid HydroShare credentials and four basic metadata values to describe their sciunit: "abstract", "title", "keywords", and "make resource public or private." In the current implementation, the resource is published on HydroShare as a generic resource type that includes just these four metadata terms. Once the resource for the sciunit is created within HydroShare, the user can log into HydroShare and edit the metadata fields to more fully describe the sciunit resource.

In addition to integrating with HydroShare for storing and publishing a sciunit, this research has also resulted in approaches for using cloud resources for execution of sciunits directly through the HydroShare user interface. We evaluated three cloud computing services to provide

this functionality: EarthCube Integration and Testing Environment (ECITE), CyVerse and Amazon Web Services (AWS). ECITE and CyVerse are both projects funded by the United States National Science Foundation (NSF) for hosting computing environments and both are currently under active development. One main advantage for using ECITE or CyVerse is that both are free of charge for scientific studies. AWS, though typically not free, does offer a competitive grant program for researchers that could also provide free resources for scientific research. While all three approaches were tested and are discussed further in Section 5, AWS was selected as the initial environment for further development for the reasons described in the Discussion and Conclusions section.

Finally, a method for integrating the cloud-based sciunit execution from the HydroShare user interface was designed and implemented. A HydroShare web-app was used for this purpose. This web-app directs users to a AWS Elastic Cloud Compute (EC2) instance where sciunits can be executed. The web-app configured to run a particular sciunit can be accessed through the "Open with…" button on the landing page for the resource that stores the raw input data. When the scientist clicks on the web-app button from the "Open with…" menu, an HTTP request containing the raw input data's resource ID will be sent to the server. With the resource ID, the HydroShare REST API, can be used to download the raw input data and the sciunit to the server. The server can then, execute the sciunit using the raw data, and return back the output to the scientist in a new HydroShare resource.

## 4.4 Example Application

A use case centered around the MODFLOW-NWT model was created to demonstrate the capability of GeoTrust *Sciunit-CLI* tool for packaging and publishing a workflow in HydroShare. The capability of then executing the packaged workflow through HydroShare is also demonstrated

focusing on the use of an EC2 instances from AWS. We used a Linux-based micro-sized machine (t2) with 1 Gb of memory, a 3 vCPU, 32 Gb of Solid State Drive (SSD)-based local instance storage, and a 64-bit platform ("Amazon EC2 Instances," 2015) for prototyping and demonstration purposes.

The first step in the process is to install the *Sciunit-CLI* on the EC2 instance. This was accomplished by completing the following steps. 1) The developer creates accounts on both Globus and HydroShare. A Globus account is needed because *Sciunit-CLI* supports Globus as a secure and efficient data transfer protocol and a HydroShare account is needed so users can post sciunits to the HydroShare system. 2) The developer downloads and installs the *Sciunit-CLI* tool following the steps detailed in the GeoTrust tutorial (http://www.geotrusthub.org/geotrust_html/GeoTrust.html). 3) At a terminal, the developer starts *Sciunit-CLI* and, if this is the first time using the application, provides the Globus credentials. This information will be stored for future uses of the application. The HydroShare credentials are requested only when the user is uploading resources to HydroShare and not when the *Sciunit-CLI* application is initiated.

Next, a workflow was created to prepare the input data for the MODFLOW-NWT model and to run the MODFLOW-NWT engine using the prepared files. In the use case, the model inputs are for simulation of the shallow groundwater flow system of the James River watershed upstream of Richmond, VA. The model simulates recharge to the water table, subsurface flow through the saturated zone, and base-flow discharge to surface water bodies including the James, Rivanna, and Hardware Rivers and several smaller-order streams. All MODFLOW input files were constructed by creating pre-processing scripts using the FloPy Python library (Bakker et al., 2016). The shallow groundwater system was simulated with a single layer of 300 m x 300 m grid cells

(approximately 260,000 total model cells). Depth-integrated effective transmissivity was assumed to be constant throughout the active model area and assigned as 100 $m^2$/day. Spatially-distributed recharge was derived from the national recharge dataset developed by Reitz et al. (2017). Base flow discharge was simulated using the MODFLOW Drain Package with all drain elevations (i.e., the water table elevation required to discharge base-flow to a receiving stream) extracted from the National Elevation Dataset.

After the workflow to pre-process the input data and run the MODFLOW-NWT model was created, it was packaged into a sciunit using the *Sciunit-CLI* tool. Figure 4.4 outlines the first steps in this process. The developer starts the tool and then uses the package command to run the workflow. This package command traces all dependencies for the workflow and includes them in a single Docker file. Figure 4.5 shows how after the tool packages the workflow and provides a "package_hash" as the unique identifier for the package. Figure 4.6 shows how the scientist uses this "package_hash" to share the package through HydroShare as a resource. If this is the first time connecting to HydroShare, *Sciunit-CLI* will ask for HydroShare credentials, otherwise the credentials stored when first input will be used. The user is then asked for four basic metadata values to describe the resource: "abstract", "title", "keywords", and "make resource public or private." Additional metadata can be provided by the user via HydroShare Graphical User Interface (GUI) and future implementations of the *Sciunit-CLI* may this functionality by automatically populating more detailed metadata for describing the resources.

```
ubuntu@ip-172-31-25-113:~/test$ sudo sciunit-cli.py
Enter "stop" to end session
Commands start with  --
undefined > --sciunit start modflow-nwt
modflow-nwt > --package workflow.sh
  > 0...10...20...30...40...50...60...70...80...90...100 - done.
  > Creating output file that is 5957P x 4395L.
```

**Figure 4.4** The process taken to start and package the workflow on linux environment using GeoTrust *Sciunit-CLI* tool

```
> Elapsed run time: 4.390 Seconds
> Normal termination of simulation
package_hash= 929d7449525059bf0e46b047ec11c51ce6ff6186
Annotated
modflow-nwt >
```

**Figure 4.5** *Sciunit-CLI* creates a package hash for the packaged workflow.

```
modflow-nwt > --publish 929d7449525059bf0e46b047ec11c51ce6ff6186
enter abstract: This resource includes a packaged workflow created by the GeoTrsut Sciunit-CLI tool.
This workflow is used for preparing the input data for  MODFLOW-NWT model and the model engine.
title: MODFLOW-NWT
keywords (split with comma):  MODFLOW-NWT,sciuint, MODFLOW-NWT Engine
do you want to make this resource public (y/n): y
resources was created
modflow-nwt > --stop
done
ubuntu@ip-172-31-25-113:~/test$
```

**Figure 4.6** The package hash is used to publish a package to HydroShare.

Figure 4.7 shows how the published resource appears in HydroShare. The resource is uploaded to HydroShare as a Composite resource type. This resource type allows the resource to include multiple files without file format limitations and with metadata associated at a file level within the resource. The composite resource contains two files. The first is the provenance metadata file created while packaging the workflow. The provenance metadata for this package

contains information concerning the creation and version history of the managed data. The second file is the zipped package for the sciunit itself.

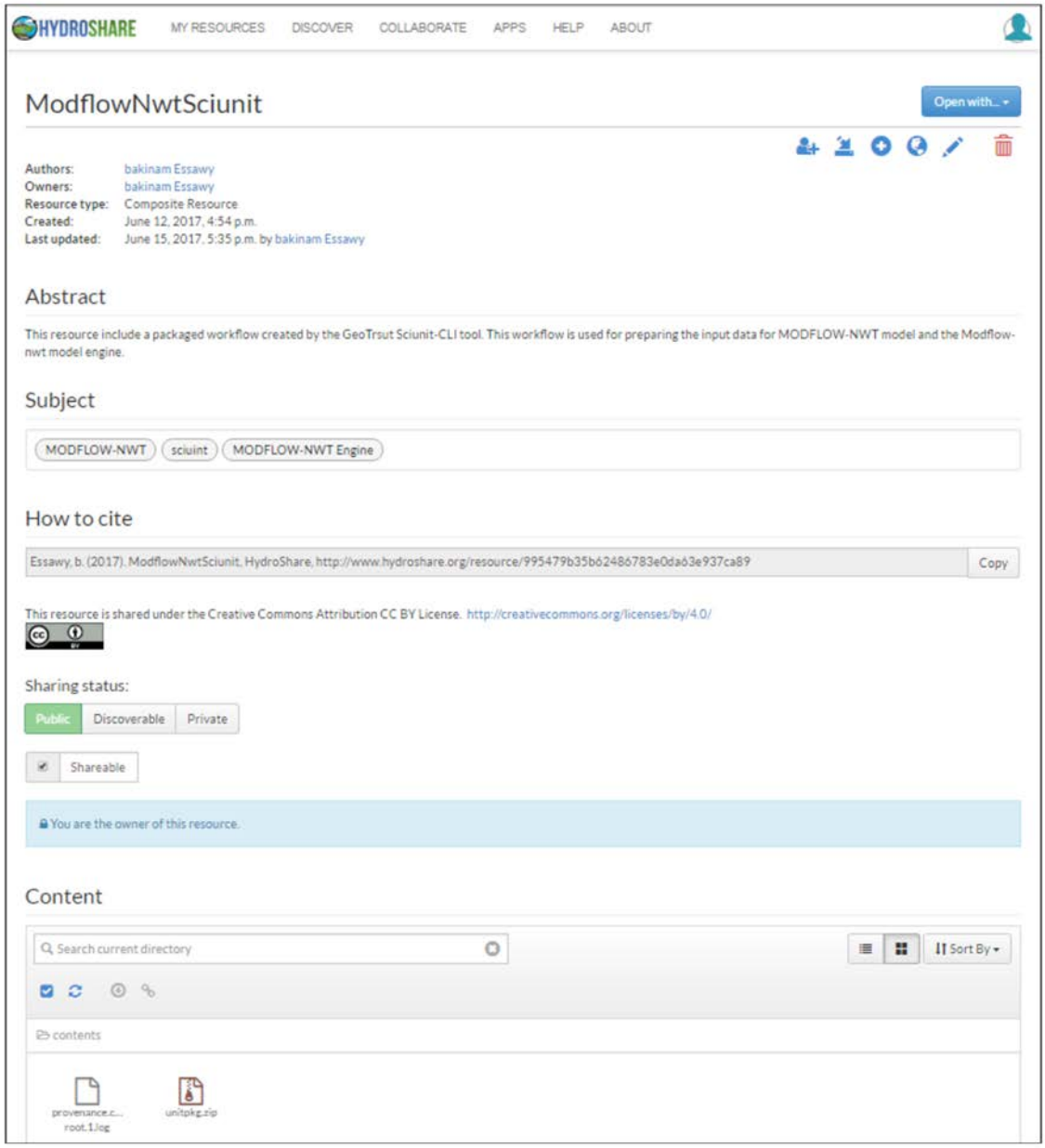



**Figure 4.7** The MODFLOW-NWT preprocessing and model engine packaged workflow published on HydroShare as composite resource.

Once the sciunit is a HydroShare resource, the scientist can use HydroShare's integration with third-party web apps to execute the sciunit with other raw data. We used the procedure

outlined in Figure 4.8 to create a new resource in HydroShare for storing the input data required by the sciunit container. We made this resource using the model instance resource type and named it "ModflowNwtRawData." Using the same procedure, we created another resource with the web-app resource type and named it "GeoTrust." This web-app points to the AWS-EC2 instance where the *Sciunit-CLI* software is installed. The GeoTrust web-app resource is linked to the ModflowNwtRawData resource by the SupportedResourceType metadata property. This allows the web-app to appear in a drop-down list when a user clicks on the "Open with" button on the ModflowNwtRawData resource landing page (Figure 4.9). When a user selects this option, the sciunit container is executed on the AWS-EC2 instance and the results are written back to HydroShare as a new resource with a MODFLOW Model Instance resource type. This resource type is used because the resource can be executed by a MODFLOW model program and it allows for adding extended metadata specific to MODFLOW (Morsy et al., 2017).

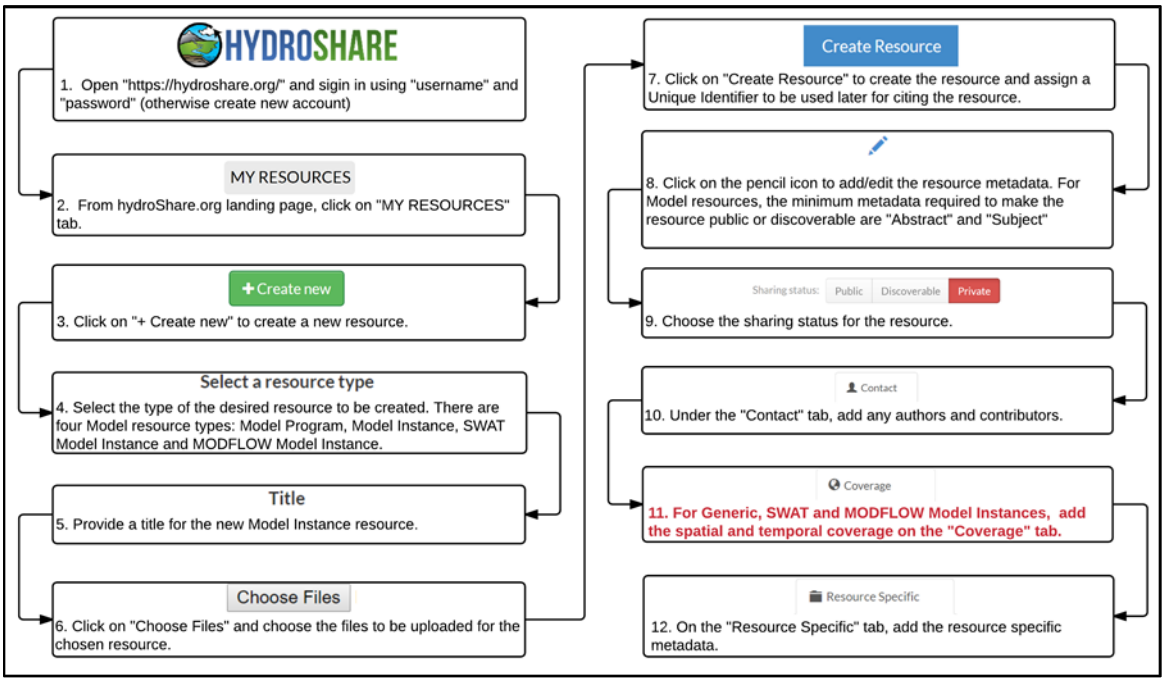



**Figure 4.8** The activity diagram showing the steps used to the create new model resource on HydroShare (adopted from Morsy et al. 2017).

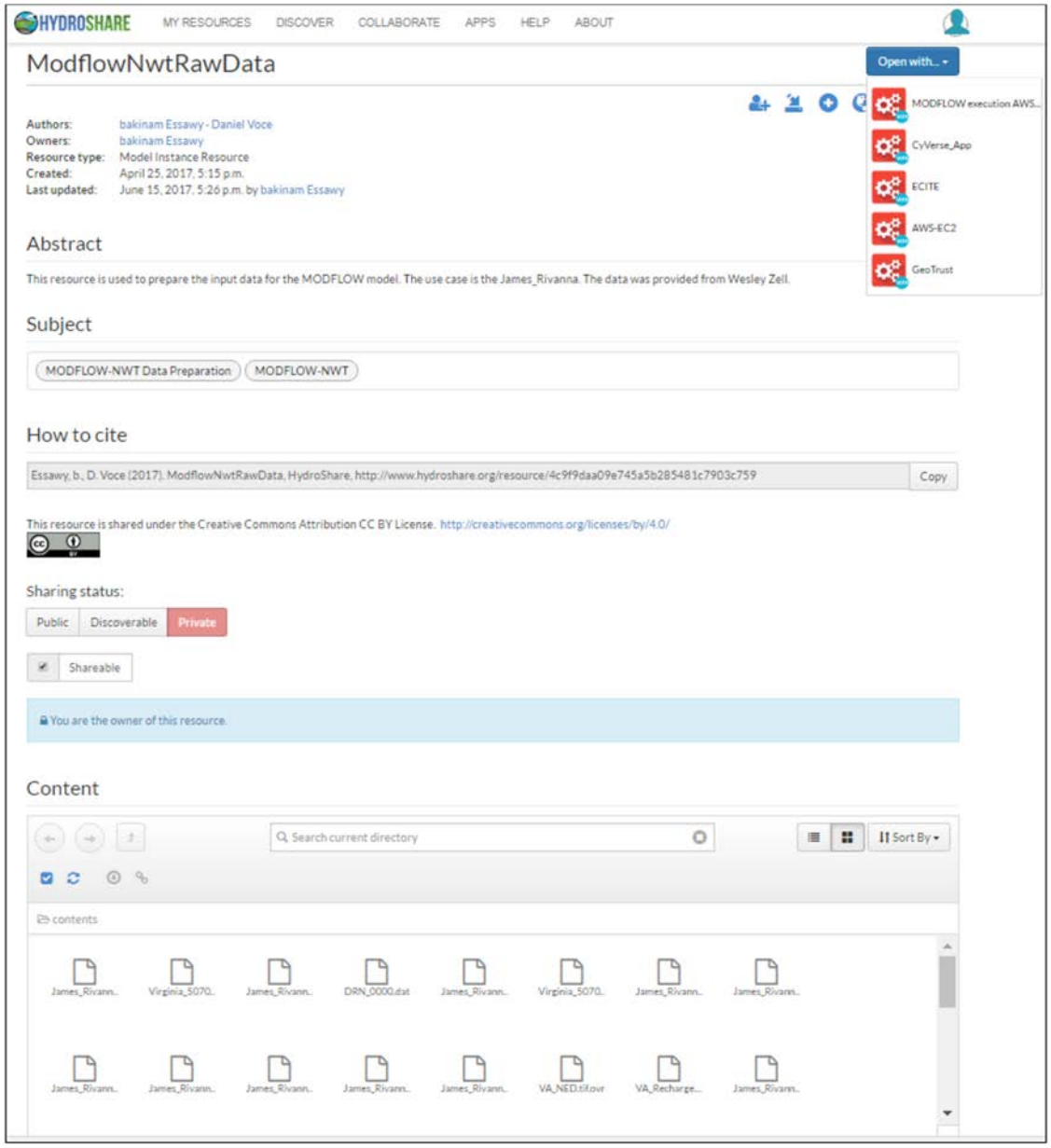



**Figure 4.9** The raw data within the model instance resource, and the web apps linked to this resource type.

Figure 4.10 presents the steps that occur when the "Open with" button is clicked on the ModflowNwtRawData resource landing page. The "Open with" app will perform a HTTP GET request to the AWS-EC2 machine, which has already been configured with the *Sciunit-CLI* following the steps described earlier. The webserver running on the AWS-EC2 machine handles the HTTP request AND automatically executes a Python script that first uses the HydroShare user

authentication to download the raw data from the ModflowNwtRawData resource and downloads the ModflowNwtSciunit resource that includes the sciunit container. Once the ModflowNwtSciunit resource is downloaded, the resource is unzipped and moved to the working directory for the analysis. *Sciunit-CLI* executes the downloaded package that prepares the raw input data for the MODFLOW-NWT model and uses this processed data to execute the MODFLOW-NWT model program itself using the processed data as input. After the model is executed, a new resource is created in HydroShare with the MODFLOW Model Instance resource type named ModflowNwtSciunitOutput and the output from the *Sciunit-CLI* execution is uploaded into this new resource. A new collection resource is also created on HydroShare to group the ModflowNwtRawData generic model instance resource (the resource type is a generic model instance since the data uploaded have no specific metadata or format that could be tied to a specific resource type), the web-app GeoTrust resource, the ModflowNwtSciunit MODFLOW model instance resource, the ModflowNwtSciunit composite resource, and the ModflowNwtSciunitOutput resource that includes the output resulting from executing the sciunit package.

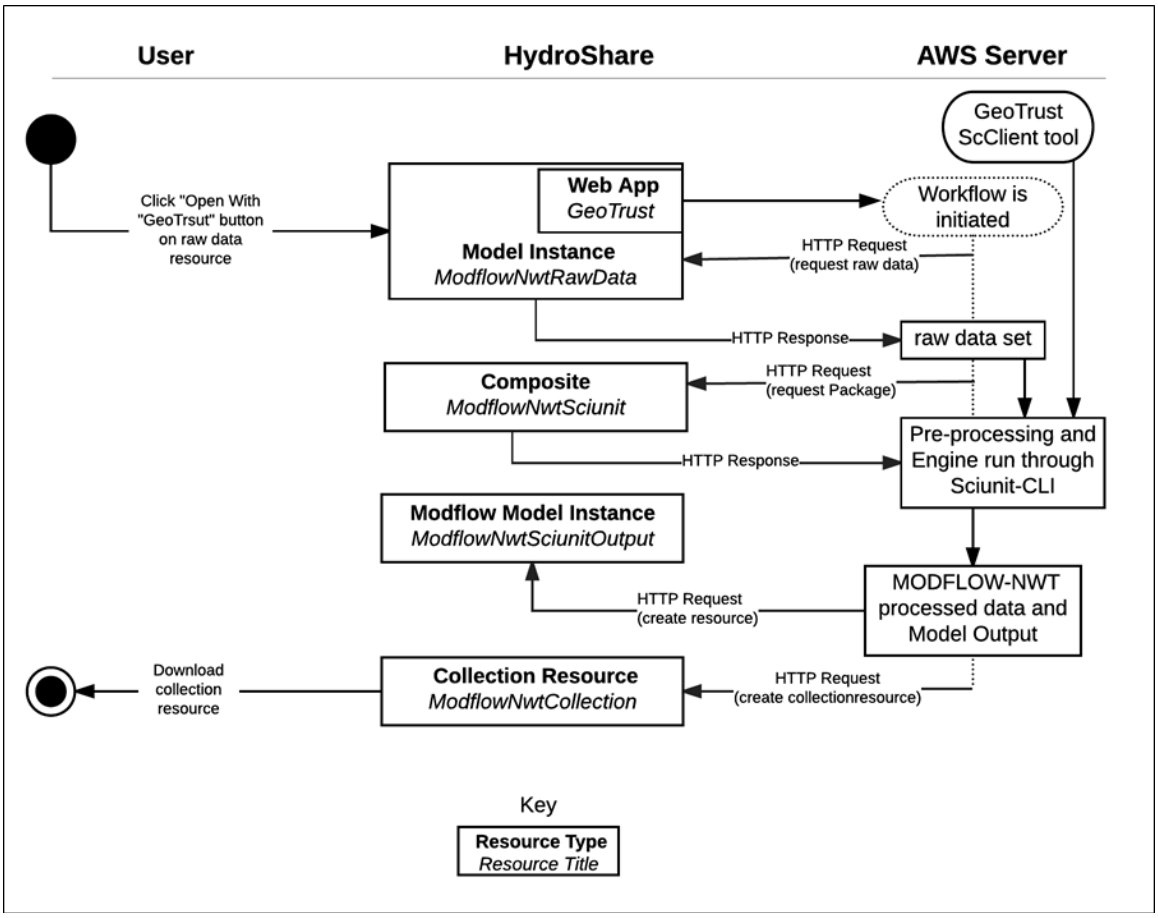



**Figure 4.10** Activity diagram showing the steps for the online execution of the sciunit through HydroShare.

Figure 4.11 shows the resources in HydroShare after using the "Open with" action on the ModflowNwtRawData resource. Two new resources are created. The first resource in the workflow is the ModflowNwtSciunitOutput resource that includes the input files for the MODFLOW-NWT model program which were prepared through the preprocessing script; it is given the MODFLOW model instance resource type. Because this resource has the inputs that are required by MODFLOW-NWT model. This allows the resource to have extended metadata specific to MODFLOW models. The second resource created is the ModflowNwtCollection resource that includes all the resources that were used in the online execution for MODFLOW-

NWT. This provides a grouping of resources used for an analysis and allows the user to share or download this collection of resources more easily.

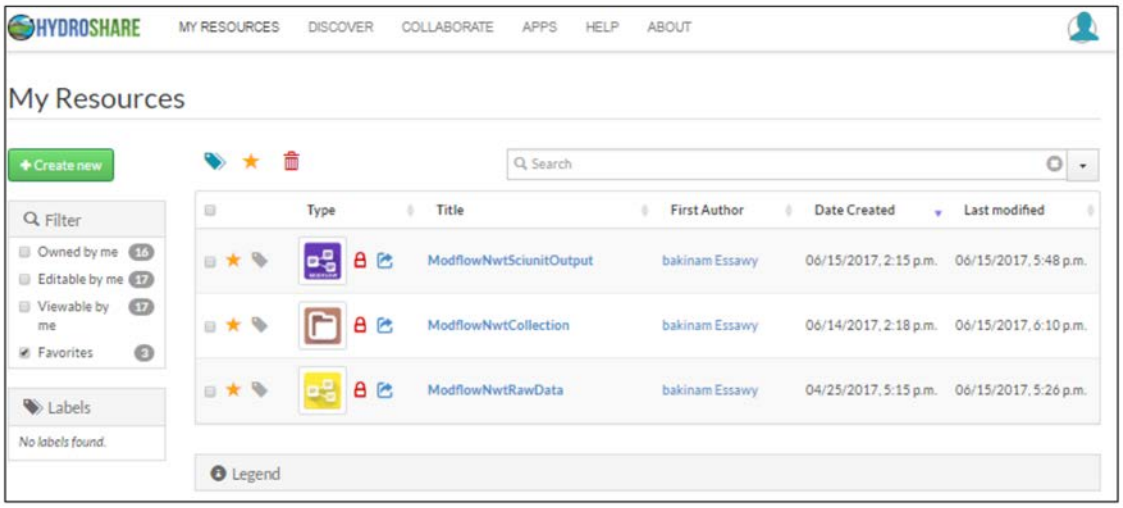



**Figure 4.11** HydroShare user My Resource page after using the GeoTrust web app for the online execution.

Figure 4.12 shows details for the ModflowNwtSciunitOutput resource as viewed on this resource's HydroShare landing page. The resource contains the output generated from running the sciunit that prepares the model input for the MODFLOW-NWT and the output from running the MODFLOW-NWT model program itself, it allows for extended metadata terms specific to MODFLOW including the concept of (packages) associated with the model. Because this resource has the MODFLOW Model Instance resource type. In this case, the MODFLOW model has eight packages. The output control (oc) package specifies how the model output is written. The discretization (dis) and basic (bas) packages define the spatial and temporal framework of the model (e.g., location of the active, inactive, and specified head cells). The upstream-weighting (upw) groundwater flow package used in the model describes the system properties (e.g., transmissivity/conductivity). The Newton-Raphson solver (nwt) package defines the model solver and its specifications. The drain package (drn) specifies the method of simulating the discharge of groundwater as base-flow in streams in rivers. The MODFLOW-NWT uses these packages and generates one output listing file (list) that contains all the information about the current run (e.g., stress period, time step, and the number of active, and inactive cells, the recharge, drains, and any errors). Finally, the name file (nam) specifies the name of the input and output files for the model instance.

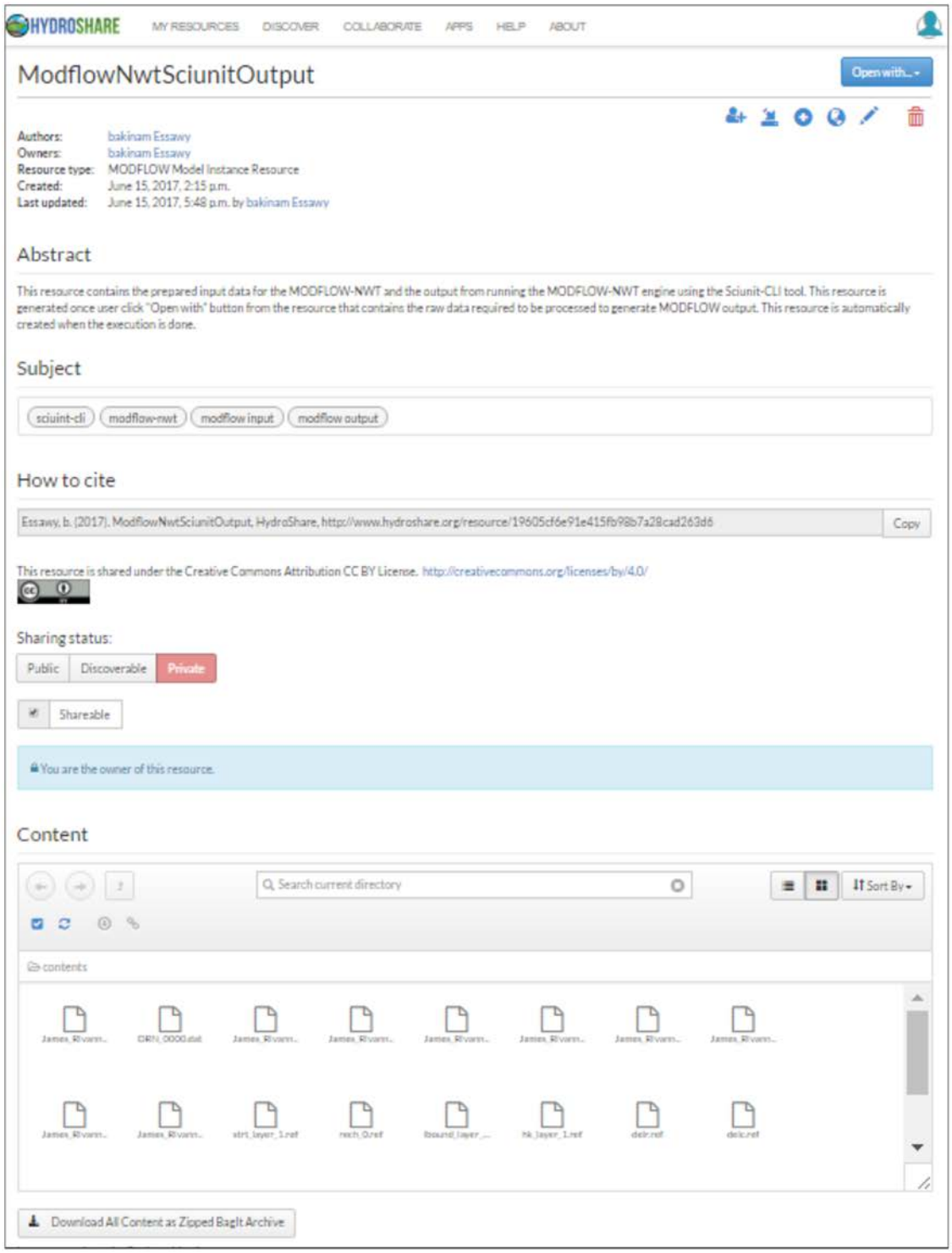



**Figure 4.12** The ModflowNwtSciuintOutput resource landing page in HydroShare.

Additional metadata associated with the MODFLOW output resource that appears on the Hydroshare landing page is divided into four categories: 1) Authorship, 2) Related Resources, 3) Resource Specific, and 4) Web Apps. Figure 4.13 shows the "Related Resources" metadata. Here, all resources linked to the MODFLOW output resource through formal relationships are listed. In this case, the MODFLOW output resource is linked to the ModflowNwtRawData resource through the "derived from" relationship and to the MODFLOW-NWT resource through the "executed by" relationship. Figure 4.14 shows the "Resource Specific" metadata. These are non-null metadata terms that apply only to the MODFLOW output's resource type (MODFLOW Model Instance) such as grid attributes and solver and boundary condition package choices. Additional metadata terms not previously populated by the user can be populated later within the edit mode and will appear in this section once populated.

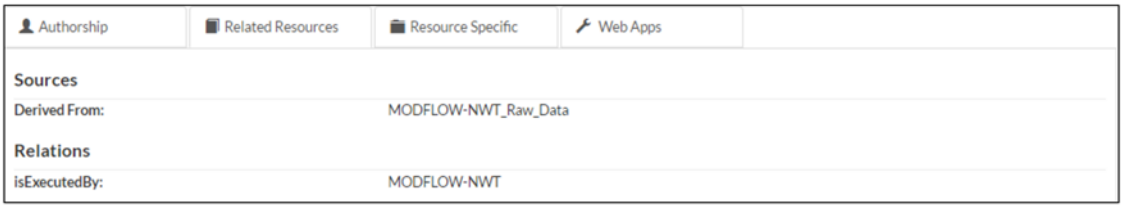



**Figure 4.13** The ModflowNwtSciuintOutput Related Resources metadata tracking the resource's provenance within HydroShare.

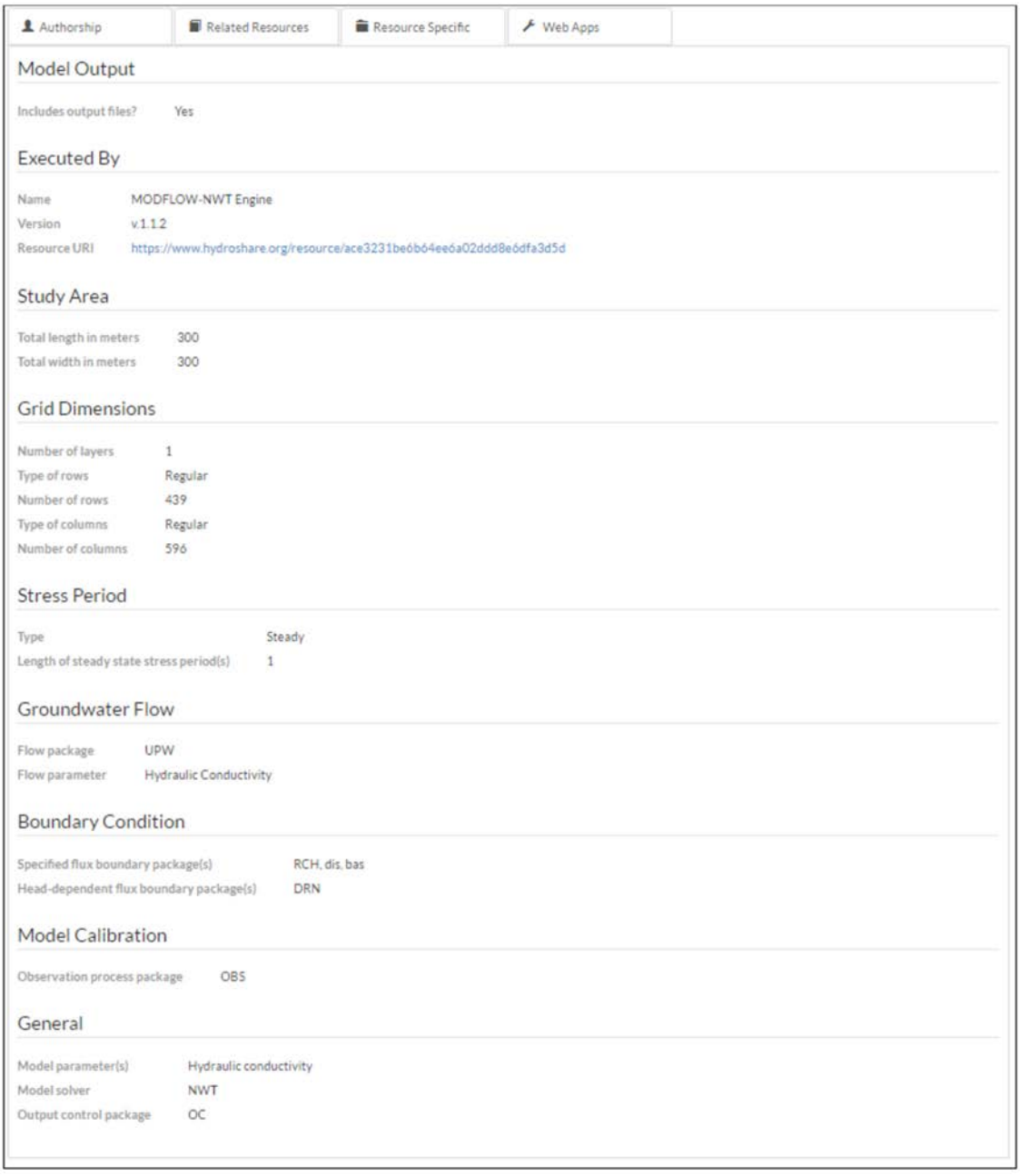



**Figure 4.14** ModflowNwtSciuintOutput specific metadata capturing key, MODFLOW-specific model properties.

Figure 4.15 shows details for the resulting ModflowNwtCollection resource as viewed on this resource's landing page. The collection resource contains four sub-resources: 1) the ModflowNwtRawData resource with the raw input data needed to be prepared for the MODFLOW-NWT model engine, 2) the ModflowNwtSciunit resource with the sciunit pre-processing workflow that also includes running the MODFLOW-NWT model program, and 3) the ModflowNwtSciunitOutput resource which stores the output generated from running the Sciunt workflow, and 4) the GeoTrust web app used to perform the online model execution using AWS-EC2. By organizing all of these resources into a single collection, it is possible to have one landing page where users can, referring back to the introduction of this study, view, obtain, and execute (1) raw initial datasets, (2) data preprocessing scripts used to clean and organize the data, (3) model inputs, (4) model results, and (5) the specific model code along with of all its dependencies used for a computational analysis.

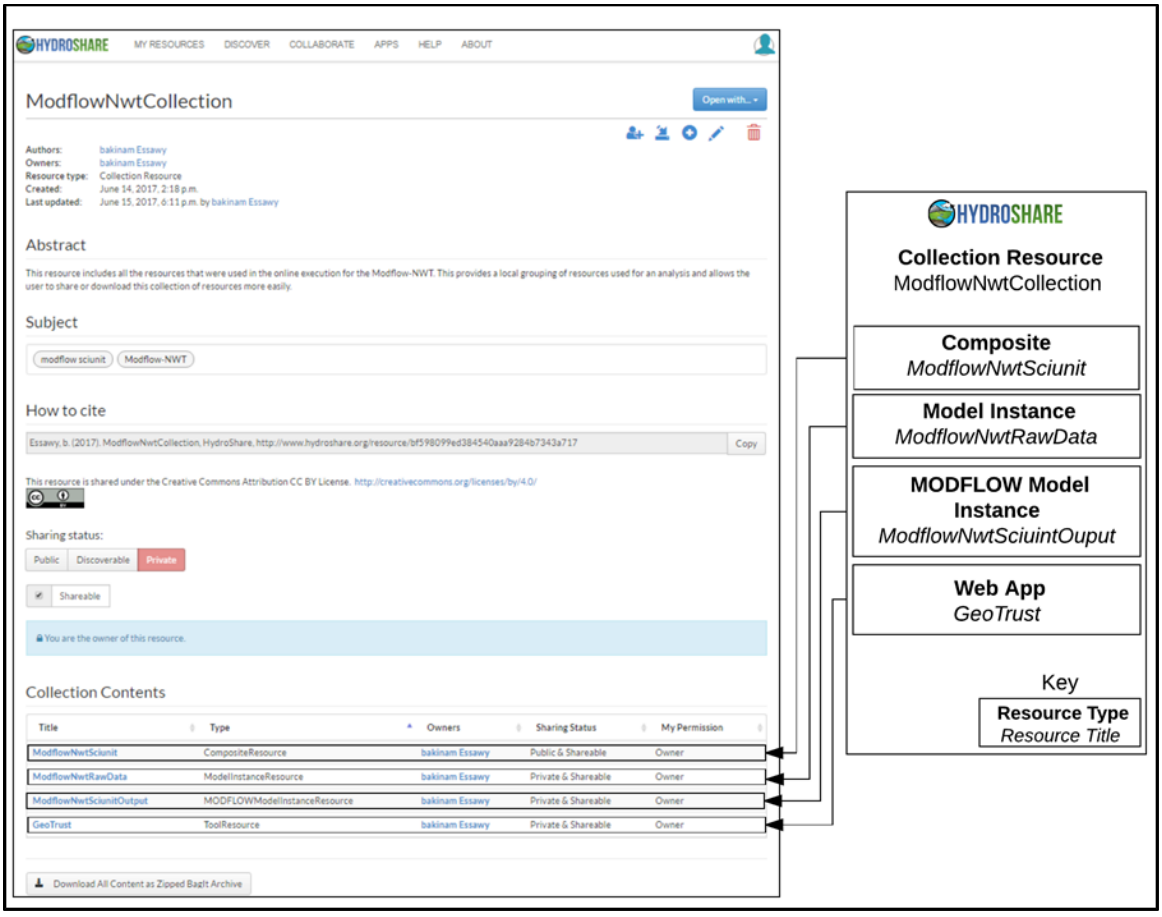



**Figure 4.15** The collection resource that includes all resources used within the study.

## 4.5 Discussion and Conclusions

In this study, we demonstrated how Hydroshare and GeoTrust can be integrated to easily and efficiently reproduce model workflows. MODLFOW-NWT was used as an example application to demonstrate the functionality provided by these cyberinfrastructures for creating open, reusable data analysis and model execution services. The approach showed how containers built using GeoTrust tools can be shared as Hydroshare resources. A cloud-based service was created to automatically retrieve raw input data from HydroShare, execute a sciunit container that prepares the MODFLOW-NWT input data and runs the MODFLOW-NWT model program, and share the results on HydroShare using a MODFLOW Model Instance resource type. All of the

resources are aggregated in HydoShare into one collection resource with domain-specific metadata.

This research demonstrates how the integration of scientific cyberinfrastructure in this case the HydroShare and GeoTrust projects, can improve reproducibility in computational hydrology. This particular integration can have practical uses such as saving time in running a MODFLOW model by not having to install local software dependencies. Hydrologists can even build a new MODFLOW model directly from raw input data (e.g., land surface DEMs or stream network shapfiles) by running a sciunit container that includes automated data preparation steps implemented using a MODFLOW Python package. The container is run online using AWS resources directly through the HydroShare user interface. A particular advantage of this approach is that the GeoTrust *Sciunit-CLI* tool allows scientists a method for efficiently creating containers for script-driven modeling workflows. Thus, the general approach demonstrated here for MODFLOW-NWT could be expanded for any workflow that can be automated and that is compatible with Docker requirements. For example, in prior work we have constructed pre- and post- processing workflows for the Variable Infiltration Capacity (VIC) hydrologic model (Liang et al., 1996b) that could directly benefit from this method for curating, packaging, and sharing resources (Billah et al., 2016c; Essawy et al., 2016)). These containers are efficient, lightweight, self-contained packages of computational experiments that can be nearly guaranteed to be repeated or reproduced regardless of deployment issues. Combining this functionality with HydroShare provides the ability to share this functionality more broadly within the hydrology community.

From our experience, AWS made the process of obtaining compute resources the simplest compared to ECITE and CyVerse, the other two cloud-based platforms tested in this research. The AWS user simply logs in to the console, selects the type of the machine needed, and launches it.

CyVerse has their own console where the user logs in and requests an instance, giving a summary of the project. The user is working on. When using ECITE, we had to contact the developer and ask for an instance with the required specifications and a short paragraph summarizing the project we are working on to justify the allocation of compute resources. With ECITE, we also needed to contact the developer each time we wanted to open a port (e.g., port 22 to ssh for or port 80 for HTTP). The service does not currently support Elastic IPs like AWS, so each time we restarted an instance and wanted to use ssh to access to the machine, we needed to report the IP address used to access the machine to the developer to add this address to the security rules. CyVerse is a more mature service, but allows each user only a certain allocation of computational time. Once the user exceeds this allocation, the instance is suspended and the user needs to request more time from the administrators. This feature was problematic for our use case of a continually available cloud-based resource for online model execution. For these reasons, we used AWS-EC2 for much of the testing work described in this study, but ECITE an CyVerse are in active development and will likely be good options for this use case in the future.

While this approach shows great promise, it is not without limitations that should be the focus of future research. Two limitations to highlight are (1) the *Sciunit-CLI* tool must be installed locally in order to re-execute a sciunit container and (2) HydroShare lacks methods for uniquely identifying and managing Web-app resources that will be needed as the number of these resources continues to increase. Regarding the latter limitation, without a more organized structure, naming conflicts could cause confusion in the "Open with" button which app to be used. Also, this work does not fully explore computational challenges associated with the proposed methodology. Using cloud services like AWS provides the opportunity for scalability as more users are added. For example, this solution used a small EC2 instances for prototyping and proof of concept

implementations. Future work could explore AWS EC2 Container Service (ECS) as an alternative for a more scalable solution to support multiple concurrent users. Data movement between HydroShare and AWS is another potential issue as data volumes increase, which is not uncommon for hydrologic modeling. HydroShare is built on iRODS (Integrated Rule-Oriented Data System), which includes the ability to interface with AWS S3 storage resources. Future work could explore using this functionality to automate the movement of large files between HydroShare and AWS to support computation within AWS and still maintain access through the HydroShare user interface. iRODS is specifically designed to handle such data federation needs and should provide a robust solution for managing the large data flows common in hydrologic modeling.

## 4.6 References


Amazon EC2 Instances [WWW Document], 2015. URL http://aws.amazon.com/ec2/instance-types/ (accessed 6.7.15).

Bakker, M., Post, V., Langevin, C.D., Hughes, J.D., White, J.T., Starn, J.J., Fienen, M.N., 2016. Scripting MODFLOW Model Development Using Python and FloPy. Groundwater 54, 733–739. doi:10.1111/gwat.12413

Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016a. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environ. Model. {&} Softw. 78, 31–39.

Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016b. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environ. Model. Softw. 78, 31–39.

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

Borgman, C.L., 2012. The conundrum of sharing research data. J. Am. Soc. Inf. Sci. Technol. 63, 1059–1078.

Cassey, P., Blackburn, T.M., 2006. Reproducibility and Repeatability in Ecology. Bioscience 56, 958–959.

David, C.H., Famiglietti, J.S., Yang, Z.-L., Habets, F., Maidment, D.R., 2016. A decade of RAPID—Reflections on the development of an open source geoscience code. Earth Sp. Sci. 226–244. doi:10.1002/2014EA000014.Received

David, C.H., Gil, Y., Duffy, C.J., Peckham, S.D., Venayagamoorthy, S.K., 2016. An introduction to the special issue on geoscience papers of the future. Earth Sp. Sci. doi:10.1002/2016EA000201.Received

De Roure, D., Goble, C., Stevens, R., 2009. The design and realisation of the Virtual Research Environment for social sharing of workflows. Futur. Gener. Comput. Syst. 25, 561–567. doi:10.1016/j.future.2008.06.010

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. doi:10.1002/2015EA000139

Fulweiler, R.W., Emery, H.E., Maguire., T.J., 2016. A workflow for reproducing mean benthic gas fluxes. Earth Sp. Sci. 3, 318–325. doi:10.1002/2015EA000158

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016a. Towards the Geoscience Paper of the Future : Best Practices for Documenting and Sharing Research from Data to Software to Provenance. Earth Sp. Sci. 1–75. doi:10.1002/2015EA000136

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J., 2007. Examining the challenges of scientific workflows. Ieee Comput. 40, 26–34. doi:10.1109/MC.2007.421

Gil, Y., Garijo, D., Mishra, S., Ratnakar, V., 2016b. OntoSoft : A Distributed Semantic Registry for Scientific Software. Proc. Twelfth IEEE Conf. eScience, Balt. MD.

Gil, Y., Ratnakar, V., Ca, R., Garijo, D., 2015. OntoSoft : Capturing Scientific Software Metadata, in: Eighth ACM International Conference on Knowledge Capture, Palisades, NY, 2015.

Gorgolewski, K.J., Poldrack, R.A., 2016. A Practical Guide for Improving Transparency and Reproducibility in Neuroimaging Research. PLoS Biol. 14, 1–13. doi:10.1371/journal.pbio.1002506

Heidorn, P.B., 2008. Shedding Light on the Dark Data in the Long Tail of Science. Libr. Trends 57, 280–299. doi:10.1353/lib.0.0036

Higgins, S., 2007. Using Metadata Standards [WWW Document]. Digit. Curation Cent. URL http://www.dcc.ac.uk/resources/briefing-papers/standards-watch-papers/using-metadata-standards#2 (accessed 5.10.16).

Horsburgh, J.S., Morsy, M.M., Castronova, A.M., Goodall, J.L., Gan, T., Yi, H., Stealey, M.J., Tarboton, D.G., 2015. Hydroshare: Sharing diverse environmental data types and models as social objects with application to the hydrology domain. JAWRA J. Am. Water Resour. Assoc. 52. doi: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. 50. doi:10.1002/2016WR019285

JB, G., PJ, G., SJ., W., 2007. OpenMI: Open modelling interface. J. Hydroinformatics 9, 175–191.

Liang, X., Lettenmaier, D.P., Wood, E.F., 1996a. One-dimensional statistical dynamic representation of subgrid spatial variability of precipitation in the two-layer variable infiltration capacity model. J. Geophys. Res. Atmos. 101(D16), 21403–21422.

Liang, X., Lettenmaier, D.P., Wood, E.F., 1996b. One-dimensional statistical dynamic representation of subgrid spatial variability of precipitation in the two-layer variable infiltration capacity model. J. Geophys. Res. Atmos. 101(D16), 21403–21422.

Liang, X., Wood, E.F., Lettenmaier, D.P., 1996c. Surface soil moisture parameterization of the VIC-2L model: Evaluation and modification. Glob. Planet. Change 13, 195–206. doi:10.1016/0921-8181(95)00046-1

Lud, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., 2006. Scientific workflowmanagement and the Kepler system. Concurr. Comput. Pract. Exp. 18, 1039–1065. doi:10.1002/cpe.994

Malik, T., 2017. GeoTrust: Improving Sharing and Reproducibility of Geoscience Applications [WWW Document]. EOL Semin. Ser. URL https://www2.ucar.edu/for-staff/daily/announcement-calendar-event/eol-seminar-series-dr-tanu-malik (accessed 6.6.17).

Mcdougal, R.A., Bulanova, A.S., Lytton, W.W., 2016. Reproducibility in Computational Neuroscience Models and Simulations. IEEE Trans. Biomed. Eng. 63, 2021–2035.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014a. Metadata for Describing Water Models, in: In Proceedings of the 7th International Congress on Environmental Modelling and Software, DP Ames, NWT QuinnMorsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the. pp. 978–988.

Morsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014b. Metadata for Describing Water Models, in: In Proceedings of the 7th International Congress on Environmental Modelling and Software, DP Ames, NWT QuinnMorsy, M.M., Goodall, J.L., Castronova, A.M., Bandaragoda, C., Greenberg, J., 2014. Metadata for Describing Water Models, in: In Proceedings of the. pp. 978–988.

Morsy, M.M., Goodall, J.L., 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. doi:10.1016/j.envsoft.2017.02.028

NISO, N., 2004. Understanding Metadata. Natl. Inf. Stand. Organ. 20. doi:10.1017/S0003055403000534

Niswonger, R.G., Panday, S., Motomu, I., 2011. MODFLOW-NWT , A Newton Formulation for MODFLOW-2005. U.S. Geol. Surv. Tech. Methods 6, 44.

Owsiak, M., Plociennik, M., Palak, B., Zok, T., Reux, C., Gallo, L. Di, Kalupin, D., Thomas Johnson, and M.S., 2017. Running simultaneous Kepler sessions for the parallelization of

parametric scans and optimization studies applied to complex workflows. J. Comput. Sci. 20, 103–111.

Peckham, S.D., Goodall, J.L., 2013. Computers & Geosciences Driving plug-and-play models with data from web services : A demonstration of interoperability between CSDMS and CUAHSI-HIS. Comput. Geosci. 53, 154–161. doi:10.1016/j.cageo.2012.04.019

Peckham, S.D., Hutton, E.W.H., Norris, B., 2013. A component-based approach to integrated modeling in the geosciences : The design of CSDMS. Comput. Geosci. 53, 3–12. doi:10.1016/j.cageo.2012.04.002

Peng, R.D., 2011. Reproducible research in computational science. Science. 334, 1226–1227.

Peng, R.D., Eckel, S.P., 2009. Distributed reproducible research using cached computations. Comput. Sci. Eng. 11, 28–34. doi:10.1109/MCSE.2009.6

Pope, A., 2016. Reproducibly estimating and evaluating supraglacial lake depth with Landsat 8 and other multispectral sensors. Earth Sp. Sci. 3, 176–188. doi:10.1002/2015EA000125

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

Ratnakar, V., Gil, Y., 2015. OntoSoft [WWW Document]. URL http://ontosoft.org/ontology/software/ (accessed 1.11.16).

Reitz, M., Sanford, W.E., Senay, Gabriel B., and Cazenas, J., 2017. Annual estimates of recharge, quick-flow runoff, and ET for the contiguous US using empirical regression equations, 2000-2013. U.S. Geol. Surv. data release. doi:https://doi.org/10.5066/F7PN93P0.

Reproducibility Guide - The rOpenSci Project [WWW Document], n.d. . 2017. URL http://ropensci.github.io/reproducibility-guide/sections/introduction/ (accessed 6.16.17).

Roure, D. De, Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Hull, D., Michaelides, D., Newman, D., Procter, R., Lin, Y., 2010. Towards open science : the myExperiment approach. Concurr. Comput. Pract. Exp. 22, 2335–2353. doi:10.1002/cpe

Scholten, Huub, Waveren, R.H. Van, Groot, S., Geer, F.C. Van, Wösten, J.H.M., Koeze, R.D., Noort., J.J., 2000. Good Modelling Practice in water management, in: In Paper Presented on Hydroinformatics. pp. 23–27.

Singh, V.P., Asce, F., Woolhiser, D.A., Asce, M., 2002. Mathematical Modeling of Watershed Hydrology. J. Hydrol. Eng. 7, 270–292.

Smith II, P., Malik, T., Berg-Cross, G., 2016. Rediscovering EarthCube: collaborate. Or collaborate not. There is no I. Digit. Libr. Perspect. 32, 153–191. doi:http://dx.doi.org/10.1108/DLP-09-2015-0017

Stodden, V., 2013. Resolving Irreproducibility in Empirical and Computational Research. IMS Bull. Online.

Tarboton, D.G., Horsburgh, J.S., Idaszak, R., Heard, J., Valentine, D., Couch, A., Ames, D., Goodall, J.L., Band, L., Merwade, V., Arrigo, J., Hooper, R., Maidment, D., 2014a. a Resource Centric Approach for Advancing Collaboration Through Hydrologic Data and Model Sharing. 11th Int. Conf. Hydroinformatics, HIC 2014, New York City, USA.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Band, L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D., 2014b. HydroShare: Advancing Collaboration through Hydrologic Data and Model Sharing. Int. Environ. Model. Softw. Soc. 7th Int. Congr. Environ. Model. Software, San Diego, California, USA. www. iemss. org/society/index/php/iemss-2014-proceedings. doi:10.13140/2.1.4431.6801

Yu, X., Duffy, C.J., Rousseau, A.N., Bhatt, G., Álvarez, Á.P., Charron, D., 2016. Open science in practice: Learning integrated modeling of coupled surface-subsurface flow processes from scratch. Earth Sp. Sci. 3, 190–206. doi:10.1002/2015EA000155

# Chapter 5: Conclusions

There are many challenges in reaching the ultimate goal of reproducibility in computational hydrologic modeling, especially when dealing with data-intensive modeling analyses that require a large, diverse set of input data and generate a large, diverse set of output data. Many geoscience disciplines utilize complex computational models for advancing understanding and sustainable management of Earth systems. Executing such models and their associated data pre- and post-processing routines can be challenging for a number of reasons including (1) accessing and pre-processing the large volume and variety of data required by the model, (2) post-processing large data collections generated by the model, and (3) orchestrating data processing tools, each with unique software dependencies, into workflows that can be easily reproduced and reused.

This research addressed three challenges related to improve reproducibility: 1) How can a hydrologist efficiently handle transfers of large datasets required for data processing pipelines in support of end-to-end hydrologic modeling; 2) Is it possible to provide component-level metadata for legacy hydrologic software; 3) Can we use container technology and scientific cyberinfrastructure to improve reproducibility of hydrologic modeling? Each challenge is addressed in a separate chapter.

Chapter 2 focused on addressing these challenges by leveraging the Workflow Structured Object (WSO) functionality of the Integrated Rule-Oriented Data System (iRODS) and demonstrates how it can be used to access distributed data, encapsulate hydrologic data processing as workflows, and federate with other community-driven cyberinfrastructure systems. Reproducibility requires more server-side data processing, where reference data and models are managed together, than what is common now. This is due to the large and increasing size of data

sets used by geoscientists and the growing complexity of software and software dependencies that require constrained environments to ensure reproducibility.

In this chapter, different federation approaches were used as means for providing interoperability across the variety of cyberinfrastructure systems needed for data access, analysis, modeling, and publication services. Federation approaches most often used in geoscience disciplines emphasize Web service APIs; however, to support large data sets, the community should have broader adoption of data grid federation approaches as well. The use of both approaches was demonstrated for a use case that leveraged four federated but heterogeneous cyberinfrastructure systems, DFC, TerraPop, SEAD, and via an existing connection with SEAD and DataONE.

The advantages of Web executable resources include the increased ability to share, reproduce, and collaborate on scientist-authored workflows. While the potential of scientific scripts and workflows as Web executable resources is clear, important issues remain related to managing large data and computation collections. This research successfully demonstrated an approach using data grids for addressing this challenge and for moving processing to reference data stored within data grids as a method for creating reproducible scientific workflows on large data sets.

Chapter 3 focused on another aspect of the reproducibility spectrum, referring back to the introduction of this dissertation. This aspect is having metadata for hydrologic modeling workflows. Software and data need to be shared, but so does their associated metadata. Sharing software with metadata encourages scientists to learn and build from prior work by reducing the time and effort to find and understand this prior work. Sharing software and its associated metadata complements the now commonly accepted idea of sharing data and its associated metadata.

The metadata for hydrologic models is rarely organized in machine-readable forms. This lack of formal metadata is important because it limits the ability to catalog, identify, attribute, and understand unique model software; ultimately, it hinders the ability to reproduce past computational studies. Researchers have recently proposed an ontology for scientific software metadata called OntoSoft for addressing this problem. Chapter 3 focused on evaluating the OntoSoft Ontology for capturing and sharing metadata for legacy hydrologic modeling software.

A data pre-processing software workflow used in association with the Variable Infiltration Capacity (VIC) hydrologic model was used to evaluate the OntoSoft Ontology. This was accomplished by exploring what metadata are available from online resources and how this metadata aligns with the OntoSoft Ontology. The results suggest that past efforts to document this software resulted in capturing key model metadata in unstructured files that could be formalized into a machine-readable form using the OntoSoft Ontology.

An underlying premise of this chapter is that having metadata for software, including for software at a fine-grain level, is useful for increasing transparency and reproducibility in science. Future work could test this assumption by surveying VIC users to better evaluate how metadata presented through the OntoSoft Portal increases their understanding of the VIC software, and how it influences their use and communication of the software with other researchers going forward.

Chapter 4 demonstrated how Hydroshare and GeoTrust can be integrated to more easily and efficiently reproduce model workflows. Reproducibility of computational hydrologic models is an important challenge that calls for more open and reusable code, data, and formal workflows that allow others to verify published findings. This requires an ability to document and share (1) raw initial datasets, (2) data preprocessing scripts used to clean and organize the data, (3) model inputs, (4) model results, and (5) the specific model code along with all its dependencies. The

MODLFOW-NWT groundwater model is used as an example to show the functionality provided by the GeoTrust and HydroShare cyberinfrastructure for creating open and reusable data analysis and model execution services.

The approach showed how containers built using GeoTrust tools can be shared as Hydroshare resources. A cloud-based service was created to retrieve raw input data from HydroShare, execute a sciunit container to prepare the MODFLOW-NWT input data and run the MODFLOW-NWT model program, and share the results on HydroShare using a MODFLOW Model Instance resource type. All of the resources were aggregated in HydoShare into one collection resource with domain-specific metadata.

Future work could explore using this functionality to automate the movement of large files between HydroShare and AWS to support computation within AWS and still maintain access through the HydroShare user interface. iRODS, which used to manage files in HydroShare, is specifically designed to handle such data federation needs and should provide a robust solution for managing the large data flows common in hydrologic modeling.

In conclusion, this dissertation presents new technologies and approaches to assist in moving hydrologic modeling to the ultimate goal of computational reproducibility. This was done by addressing three core challenges and by focusing on the widely used Variable Infiltration Capacity (VIC) land surface hydrologic model and MODFLOW groundwater hydrologic model. This research has resulted in approaches that leverages modern computational methods to assist hydrologists with limited knowledge in computer science to more easily reproduce their models, thus making models in the future more open, transparent, and reusable.