Cyberinfrastructure to Support Flood Modeling from Catchment to Regional Scales

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

Mohamed Morsy Anwar Morsy

August 2017

APPROVAL SHEET

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

Mohamed Morsy Anwar Morsy

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.

Michael Fitch, Ph.D.

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science
August 2017

Abstract

Flooding events are projected to increase in frequency and intensity in coming years due to climate change. New tools and approaches are needed to assist decision makers in better understanding and addressing societal impacts due to flooding and how to mitigate these impacts. This research addressed three challenges related to flooding impacts: (i) better understanding how distributed stormwater infrastructure can mitigate flooding in urban catchments, (ii) designing and building spatially-detailed, real-time flood warning systems for emergency management purposes, and (iii) designing and building cyberinfrastructure to support reuse and transparency in both flood modeling and hydrologic modeling more broadly. The goal of this research was to address these challenges by conducting three studies.

The first study explored building a catchment-scale flood model used to improve understanding of how distributing low impact development (LID) practices at the parcel level in an already urbanized watershed reduces runoff and, therefore, flood risk. At this scale and in an urban environment, spatially detailed descriptions of the physical environment are required. A physically-based modeling approach was used in order to answer "what if" hypothetical scenarios of rain garden adoption rates and their impact on watershed-scale runoff generation.

The second study explored building an automated cloud-based system for forecasting flooded roadway and bridge locations at a regional-scale. Because the study area has very low topographic relief, a two-dimensional (2D), computationally-expensive hydrodynamic model is required. This study demonstrated the ability of using instances in a public cloud with powerful graphical processing units (GPUs) to run a large (average of 4 million nodes) 2D hydrodynamic model in a time frame relevant to real-time emergency management applications. The steps required to build this system were (i) creating an automated workflow for obtaining and processing

forecast rainfall data, (ii) running the 2D model in the cloud, (iii) using geospatial analysis tools to identify flooded bridges, and (iv) presenting the results online for decision makers. The system automates forecast data access and pre-processing, execution of a high-resolution 2D hydrodynamic model, and map-based visualization of model outputs using Amazon Web Services (AWS).

The third study advanced approaches for sharing hydrologic models, such as the models created in this dissertation, through community supported cyberinfrastructure. Sharing models is important for scientific reproducibility, reuse, and fidelity. In this study, the first task was to design a metadata framework for hydrologic models that is flexible and applicable across the wide variety of models used by hydrologists. Then the study demonstrated the utility of this framework for sharing, publishing, and reusing models through an implementation within the HydroShare cyberinfrastructure system.

In the first study, the results suggest that rain gardens with 30 cm berm heights and a total area equal to 20% of the impervious surfaces within the watershed should provide sufficient storage to mitigate flooding for rain events up to and including a 10 year return period storm event. The results also suggest approximately 15%, 27%, and 38% of the runoff generated from impervious surfaces should be diverted to the rain gardens to mitigate flooding from 2, 5, and 10 year return period storm events, respectively. Given prior work on the adoption of LID approaches for other watersheds, rain gardens could effectively mitigate up to a 5 year return period storm event within the watershed, although further research on possible adoption rates in the study watershed is needed to more fully support this conclusion. In the second study, an 80x speed-up in execution time of the 2D models was achieved by using GPUs rather than a central processing unit (CPU). A prototype deployment system was built within the Amazon Web Services (AWS)

cloud that includes a web front-end, execution for the model engine, and storage of the model output data. The system is designed to run automatically during extreme weather events, produce near real-time results, and consume few computational resources until triggered by an extreme weather event. Although the model is built for a specific region of Virginia, the architecture serves as an example that could be replicated to other regions where 2D hydrodynamic models are required for real-time flood warning applications. In the third study, a general approach for representing environmental model metadata that extends the Dublin Core metadata framework was proposed. The framework was implemented within the HydroShare system and applied for a hydrologic model sharing use case. This example application demonstrates how the metadata framework implemented within HydroShare can assist in model sharing, publication, reuse, and reproducibility.

Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful, first and above all, I praise Allah for providing me this opportunity and granting me the strengths and His blessing to complete this dissertation successfully. This dissertation would never been completed without the support, assistance, and guidance of several people who mean a lot to my life. Therefore, I would like to offer my sincere thanks and appreciation to all of them.

Special thanks and appreciation goes to my supervisor, **Dr. Jonathan Goodall**, for his thoughtful guidance, encouragement, comments, and valuable advice. Really, I could not imagine having a better advisor and mentor for my Ph.D. degree. My appreciation and gratitude also goes to my committee members, **Dr. Teresa Culver**, **Dr. John Porter**, **Dr. Marty Humphrey**, and **Dr. Michael Fitch,** for serving on my Ph.D. dissertation committee and for providing valuable feedback and comments regarding my dissertation. I also would like to thank my friends, **Dr. Wesley Zell**, **Jeffrey Sadler**, **Gina O'Neil, Ben Bowes**, **Yawen Shen** and **Alex Chen,** for their continuous support and advice throughout the research.

I warmly thank and appreciate my family on their continuous encouragement, support, and the love they had for me throughout my life. I feel a deep sense of love and gratitude to my beloved parents, my father, **Dr. Morsy Anwar Morsy**, and my mother, **Reda Mohamed Elnahta**, who gave me the love and sacrificed a lot to raise me in a good environment built on love and faith. This dissertation is entirely dedicated to them. Truly, father, you are my idol and main guidance in this life and, hopefully, by achieving this, I was able now to satisfy your dreams. I also would like to thank my brothers, **Ahmed** and **Amr**, and my sisters, **Mona**, **Mena**, **Basma**, and **Heba**, for their support and encouragement to accomplish this work.

To my wife, **Dr. Bakinam Tarik Essawy**: I cannot imagine my life without you. You mean everything to me. May Allah protect and bless you for me and for our small family. **Bakinam**, my very special thanks and sincere gratitude goes to you. Thank you for your help, understanding, and tolerating the late nights and early mornings I spent to accomplish this work. I could not have accomplished this work without your love and support. We had a hard journey together to accomplish our Ph.D. and we were able to accomplish this successfully. My little princesses, **Zeina** and **Kenzy**, thanks a lot for your understanding and tolerating all the hard times that we have been through during this long process to achieve our degrees. Hopefully both of you, **Zeina** and **Kenzy**, will one day be proud of your father and what he accomplished.

Last but not least, I would also like to thank all those not mentioned who provide me with support and encouragement. Your kindness means a lot to me. I would not have made it so far in life without your support and may Allah give you all the best in return.

وَقُلِ ٱعۡمَلُواْ فَسَيَرَى ٱللَّهُ عَمَلَكُمۡ وَرَسُولُهُۥ وَٱلۡمُؤۡمِنُونَۖ وَسَتُرَدُّونَ إِلَىٰ
عَٰلِمِ ٱلۡغَيۡبِ وَٱلشَّهَٰدَةِ فَيُنَبِّئُكُم بِمَا كُنتُمۡ تَعۡمَلُونَ ١٠٥

"And say: Work; so Allah will see your work and (so will) His Messenger and the believers; and you shall be brought back to the Knower of the unseen and the seen, then He will inform you of what you did." Quran: Al-Tawba (105)

(وَقُل رَّبِّ زِدْنِي عِلْمًا)

"O my Lord! advance me in knowledge." Quran: Taa-Haa (114)

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Floods were the number one natural disaster in the US in terms of the lives lost and property damage during the 20th century (Perry, 2000). Statistics show that from 2006 to 2015, total flood insurance claims averaged more than $1.9 billion per year (NFIP Statistics, 2016). Rainfall events are predicted to become more frequent and intense due to climate change, which is expected to cause increased flooding (Melillo et al., 2014). As society faces flooding events with increasing frequency and intensity, flood modeling systems will become an even more important tool for decision makers. Such models can be used to warn municipalities and communities of forecasted flooding impacts. They can also be used to test alternative flood mitigation strategies for addressing flood problems. This research advances knowledge of flooding impacts and modeling methods by focusing on three knowledge gaps: the use of distributed stormwater controls for flood mitigation in small urban catchments, the use of 2D hydrodynamic models for flood warning at the regional-scale, and methods for documenting, sharing, and reusing flood models within the scientific and management communities.

At the catchment-scale (~10 - 100 km^2), flood models require a detailed description of the physical environment, especially for an urban watershed where stormwater infrastructure plays a significant role in the system. These models can be used for testing alternative solutions to flooding problems. For example, in the stormwater management community, low impact development (LID) approaches are drawing increased attention (Dietz, 2007). LID as a concept integrates land development and environmental concerns with the goal of minimizing the negative impacts of land development (Davis, 2005). LID approaches often emphasize distributed stormwater controls, such as rain gardens, implemented at the parcel-scale. For highly urbanized watersheds without

sufficient space for new centralized stormwater controls, LID approaches may be able to mitigate flooding impacts caused by intensive storm events.

At the regional-scale (~$10x10^3$ - $100x10^3$ km^2), flood models often face computational challenges, especially when modeling low-relief terrains that require more detailed two dimensional (2D) hydrodynamic models. For these low-relief terrains, one dimensional (1D) models are not sufficient due to the limitations of assumed uniform water velocity and constant water surface elevation modeled on each cross section (Garcia et al., 2015). Executing 2D hydrodynamic models at the regional scale requires parallel computation in order to run the model in a time frame reasonable for flood warning applications. Graphical processing units (GPUs) have recently been shown to be effective for running 2D hydrodynamic models with speed-ups of ~100x (Huxley and Syme, 2016; Garcia et al., 2015). These new computational approaches suggest that regional flood warning systems can be implemented with the spatial resolution needed to provide targeted and detailed information to decision makers.

Regardless of the scale of a modeling application, it is important to be able to share the model, its inputs, and its results with others. As demonstrated in this research, it takes a significant amount of effort to collect data, construct model inputs, and calibrate and validate model parameters. From a pragmatic perspective, this is an inefficient use of a scientist's or engineer's time. Perhaps more importantly, it inhibits the ability to reproduce or reuse studies that have a significant computational modeling component (David et al., 2016; Essawy et al., 2016; Gil et al., 2016). One way to begin to address these challenges is through better approaches for sharing and reusing models built by others. Just as there has been a major push to make better use of data collected and maintained by others, the scientific community can benefit from a similar push to

make better use of models built by others. A challenge is how to achieve model sharing given the diversity of models used within the hydrology community.

The overarching objective of this research is to advance knowledge in flood modeling and flood mitigation strategies. This objective is achieved through studies targeting three distinct but related research questions in the following three chapters. Chapter 2 addresses the first research question: "What potential does distributed green infrastructure have for flood mitigation at the catchment-scale, especially for highly urbanized catchments?" There is evidence that LID approaches distributed at the parcel-scale could have significant impact on runoff reduction at the watershed-scale, but there is a lack of agreement on the extent of this reduction, especially for flood mitigation purposes. Few, if any, studies have researched the required adoption level necessary to achieve sufficient runoff reduction to reduce flood risks within an urban watershed for storms with different return periods.

Chapter 3 addresses the second research question: "Is it possible to design and build a cloud-based regional flood system for warning and emergency management purposes for low-relief terrains?" Having the ability to accurately and quickly project potential impacts to transportation infrastructure due to forecasted weather events will become more critical given increased intensity of rainfall expected with climate change. A challenge facing this type of regional model, especially for low-relief terrains, is that it would require a computationally-expensive 2D hydrodynamic model. The ability to run a large (average of 4 million nodes) 2D hydrodynamic model in a time frame relevant to real-time emergency management applications is one example of such a challenge.

Chapter 4 includes methods and solutions for the third research question: "How should model metadata and cyberinfrastructure be designed to better support reuse and transparency in

hydrologic modeling?" Advanced approaches, through community supported cyberinfrastructure, are required for sharing hydrologic models like those created in this dissertation. Sharing models is important for scientific reproducibility, reuse, and fidelity. A motivating factor for this research is the design and development of a new system called HydroShare (https://www.hydroshare.org). The goal of HydroShare is to advance hydrologic science by enabling the scientific community to more easily and freely share products resulting from their research - not only the scientific publication summarizing a study, but also the data and models used to create the scientific publication (Horsburgh et al., 2015; Tarboton et al., 2014; Tarboton et al., 2013).

Finally, Chapter 5 provides key conclusions and suggested future research based on the research outcomes from Chapters 2 - 4.

1.1 References

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

Davis, A.P., 2005. Green Engineering Principles Promote Low-impact Development. Environ. Sci. Technol. 39, 338A–344A. doi:10.1021/es053327e

Dietz, M.E., 2007. Low impact development practices: A review of current research and recommendations for future directions. Water. Air. Soil Pollut. 186, 351–363.

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

Garcia, R., Restrepo, P., DeWeese, M., Ziemer, M., Palmer, J., Thornburg J., Lacasta, A., 2015. "Advanced GPU paralellization for two-dimensional operational river flood forecasting." In 36th IAHR World Congress.

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

2016. Towards the Geoscience Paper of the Future: Best Practices for Documenting and Sharing Research from Data to Software to Provenance. Earth and Space Science doi:10.1002/2015EA000136

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 Journal of the American Water Resources Association 52, 4. doi:10.1111/1752-1688.12363

Huxley, C., Syme, B., 2016. TUFLOW GPU – Best Practice Advice for Hydrologic and Hydraulic Model Simulations. *In: Proceedings of the 37th Hydrology and Water Resources Symposium (HWRS)*, Queenstown, New Zealand, 2016.

Melillo, J.M., Richmond, T.T., Yohe, G.W., 2014. Climate change impacts in the United States. Third National Climate Assessment.

NFIP Statistics, 2016. NFIP Statistics. The official site of the NFIP accessed December 8, 2016. https://www.floodsmart.gov/floodsmart/pages/media_resources/stats.jsp.

Perry,C. A., 2000. Significant Floods in the United States During the 20th Century - USGS Measures a Century of Floods. Kansas Water Science Center accessed December 8, 2016. https://ks.water.usgs.gov/pubs/fact-sheets/fs.024-00.html.

Tarboton, D., Idaszak, R., Horsburgh, J., Heard, J., Ames, D., Goodall, J., Band, L., Merwade, V., 2014. A Resource Centric Approach For Advancing Collaboration Through Hydrologic Data And Model Sharing. International Conference on Hydroinformatics. CUNY Academic Works. http://academicworks.cuny.edu/cc_conf_hic/314.

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

Chapter 2: Distributed Stormwater Controls for Flood Mitigation within Urbanized Watersheds: Case Study of Rocky Branch Watershed in Columbia, South Carolina[1]

2.1 Introduction

Low impact development (LID) approaches are attracting increasing interest in stormwater management (Dietz, 2007). LID as a concept integrates land development and environmental concerns with the goal of minimizing the negative impacts of land development (Davis, 2005). LID approaches differ from traditional stormwater management approaches in a number of key ways. First, they seek to minimize disturbance of a site by mimicking the natural hydrology of the site. Second, they emphasize maintaining the predevelopment runoff volume through increased infiltration of runoff generated from impervious surfaces, in contrast to the traditional approach, which focuses primarily on the mitigation of peak flow rates for larger storm events. Third, they emphasize distributed, parcel-scale controls for runoff infiltration, storage, and detention (Abi Aad et al., 2010; Dietz, 2007). One of the more commonly used LID approaches to stormwater management is bioretention technology (e.g., bioinfiltration and rain gardens) (Davis et al., 2009). For flood mitigation, which is the focus of this work, rain gardens have been found to be an effective LID approach given their ability to store and infiltrate runoff (Abi Aad et al., 2010).

Distributing LID approaches throughout a watershed could offer large benefits for highly urbanized watersheds. Many urbanized watersheds experience flooding because they were developed prior to stormwater regulations and have insufficient storage for runoff generated from

[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:
Morsy, M.M., Goodall, J.L., Shatnawi, F.M., Meadows, M.E., 2016. Distributed Stormwater Controls for Flood Mitigation within Urbanized Watersheds: Case Study of Rocky Branch Watershed in Columbia, South Carolina. *Journal of Hydrologic Engineering*, *21*(11), p.05016025.

impervious surfaces. Owing to the level of urbanization, there may simply be no space left to add large stormwater detention facilities, which greatly restricts the stormwater engineer's design flexibility. If it is possible to reduce runoff at the parcel-scale using distributed LID approaches like rain gardens, then the need for large, centralized stormwater facilities would be reduced. Because this study considers the case where LID approaches are added to an already urbanized watershed, it is referred to as a retrofit case study. The study results also apply to new construction, although new construction would allow for greater design flexibility than retrofit applications.

Most prior research on the application of LID approaches used field studies to investigate how LID approaches compare to traditional stormwater practices (e.g., Bedan and Clausen, 2009; Line et al., 2011). A common approach has been to take a watershed-scale perspective looking at paired basins where one basin adopted LID approaches for stormwater management and the other basin used more traditional stormwater best management practices (BMPs). Studies taking this approach have concluded that LID techniques are more effective at reducing runoff volumes than traditional approaches (e.g., Bedan and Clausen, 2009; Selbig and Bannerman, 2008). These studies have generally been limited to relatively small watersheds (less than 1 km^2) and to new developments. Applying field studies to larger urbanized watersheds is challenging given their size and the inability to install a large number of LID practices and measure their impact on the system. Instead, modeling approaches can be used to address the question of how LID adoption in a large, already urbanized watershed might reduce runoff volumes and flood risks.

Some modeling studies of urban watersheds have begun to look at the effectiveness of distributed stormwater controls at the watershed-scale. Damodaram et al. (2010) presented a modeling approach to incorporating LID practices into an existing hydrologic model to determine watershed-scale impacts on runoff delivered to streams. After applying the model to a watershed

in College Station, Texas, the researchers concluded that LID approaches, while effective for controlling runoff from small events, may yield little runoff control for flood events. The study watershed was large, but not as densely developed as the watershed used in this study. Also, the study did not use the latest version of the Storm Water Management Model (SWMM version 5) with new modules for simulating the impact of LID on runoff reduction. A related study using SWMM version 5 with LID modeling capabilities investigated the potential for LID techniques to mitigate projected increases in precipitation under climate change scenarios for New York City (Zahmatkesh et al., 2015). The researchers found that retrofits with LID controls could reduce average annual runoff by over 40% and peak flow rates by approximately 10%.

Field and data analysis studies have supported the claim that LID practices distributed throughout a watershed could have significant impacts on watershed-scale hydrology for highly urbanized watersheds. A study in Wilmington, North Carolina, showed stormwater control measures added as retrofits to an urbanized watershed in municipal rights-of-way was able to reduce peak discharge by 28% (Page et al., 2015). Researchers investigating a suburban watershed in Cincinnati also showed evidence that the adoption of rain gardens and barrels at the parcel level could have a significant effect on watershed hydrology (Shuster and Rhea, 2013). Eighty-five rain gardens and 174 rain barrels were installed in a 1.8 km^2 urban watershed. Even with this level of adoption, a small but significant reduction in runoff was observed between pre- and post-LID implementation conditions. The study also showed that LID practices at the parcel level could be successfully implemented with novel economic programs. Finally, Loperfido et al. (2014) analyzed observational data from different watersheds in suburban Washington, District of Columbia, and concluded that distributed stormwater controls might be an effective means of reducing runoff volumes during extreme precipitation events.

Based on these prior studies, there is evidence that LID approaches distributed at the parcel-scale could have a significant impact on runoff reduction at the watershed-scale, but there is a lack of agreement on the extent of this reduction, especially for flood mitigation purposes. Few, if any, studies have researched the required adoption level necessary to achieve sufficient runoff reduction to reduce flood risks within an urban watershed for storms with different return periods. The primary objective of this study, therefore, is to improve understanding of how the rate of adoption of LID practices at the parcel level in an already urbanized watershed affects runoff detention and, therefore, flood risk. The Rocky Branch Watershed in Columbia, South Carolina, is used as a case study for the present research. This watershed faces a recurrent flooding problem, as discussed in the "Study Area" section, that is common in many other older cities with insufficient stormwater controls. Using SWMM and a number of modeling scenarios, simulations were used to determine the LID practices that must be adopted to reduce peak flows in the watershed so as not to exceed bankfull conditions.

2.2 Study Area

The Rocky Branch Watershed is approximately 10.75 km^2 in area and is located in downtown Columbia, South Carolina (Figure 2.1). Rocky Branch is approximately 6.5 km long and discharges into the Congaree River. The watershed has long experienced recurrent flooding problems, in particular in a low-lying commercial district called Five Points (highlighted in Figure 2.1-a). Flooding typically occurs during intense summer thunderstorms. Significant efforts have been made to mitigate these flooding problems using traditional stormwater controls, but flooding still occurs at regular (approximately annual) intervals (Monk and Holleman, 2010; NOAA, 2010; Santaella and Gillbert, 2011; The State, 2012; The State, 2014; WIS TV, 2015). The headwaters

include residential communities and a portion of the University of South Carolina campus, where it would be feasible to install rain gardens as stormwater controls.

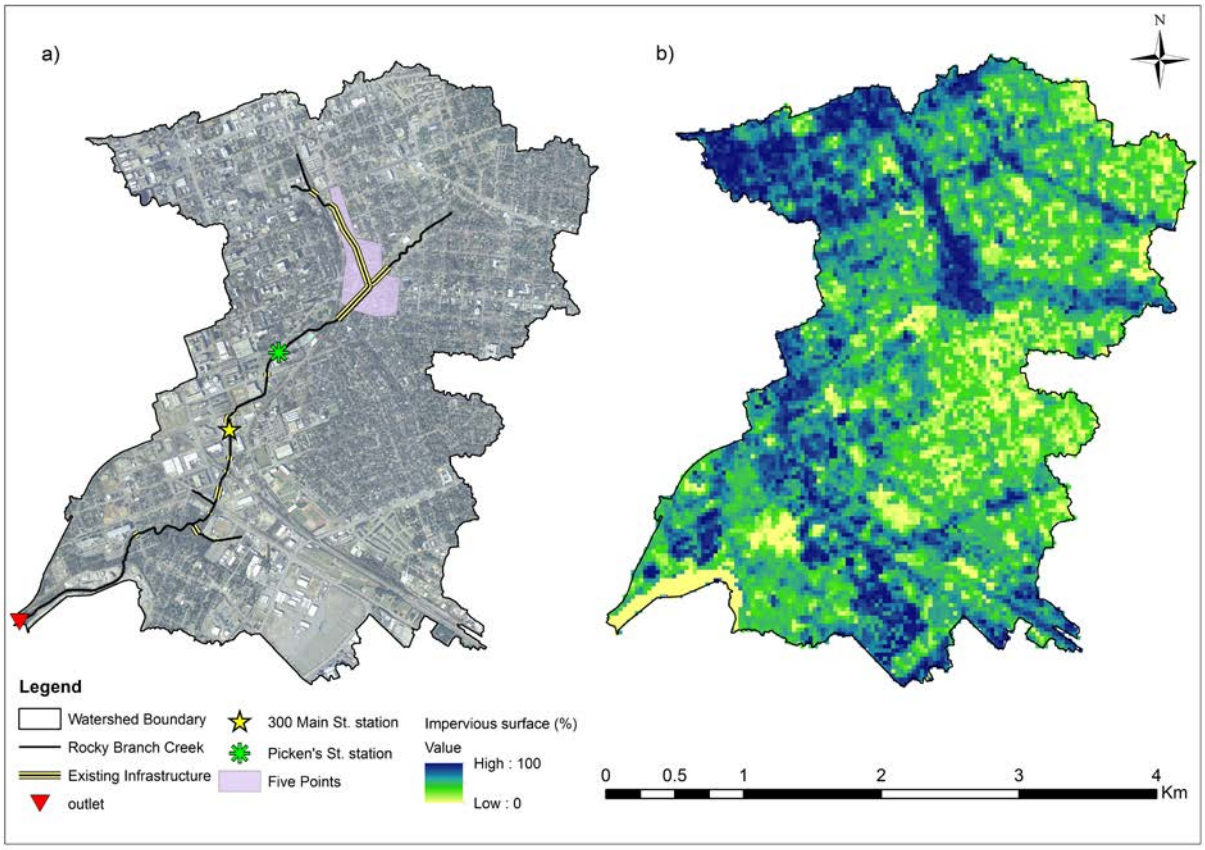


Figure 2.1 (a) Rocky Branch Watershed in downtown Columbia, South Carolina, and (b) impervious surfaces according to the National Land Cover Dataset (orthoimagery data from USGS National Map; impervious surface layer data from NLCD, 2011).

Owing to the high percentage of impervious surfaces and steep slopes, Rocky Branch has a flashy response to rainfall events. The time to peak for an observed storm event that caused flooding at the Pickens Street station (Figure 2.1) was approximately 1 hour. According to the 2011 National Land Cover Dataset (NLCD), 97% of the watershed is developed (17% high intensity, 37% medium intensity, 31% low intensity, and 12% developed open space), and much of the watershed is impervious (Figure 2.1-b). Taking just the impervious surfaces within the watershed, the maximum landscape slope is 42%, and approximately one-fifth of the landscape

has a slope greater than 5%. It is well known that impervious surfaces, and in particular connected impervious surfaces, increase runoff and flooding if not mitigated through stormwater controls and BMPs (Arnold and Gibbons, 1996; Lee and Heaney, 2003; Roesner and Urbona, 1998; Schueler, 1995).

2.3 Data and Methods

A modeling approach was used during this study to improve understanding of how the rate of adoption of LID practices at the parcel level in an urbanized watershed impacts runoff detention and, therefore, flood risk. The model selected was the SWMM developed by the U.S. Environmental Protection Agency (USEPA). A modeling approach was used because of the need to answer "what if" hypothetical scenarios of rain garden adoption rates and their impact on watershed-scale runoff generation. In this section, the SWMM model is described along with its relevance for modeling watershed-scale hydrology and role in LID adoption at the watershed-scale. Next, the steps required to prepare input data for the model are described, followed by a discussion of the model calibration and evaluation to provide confidence in the modeling results. Finally, there is a discussion of the modeling scenarios conducted to address the study research objective.

2.3.1 Model Description and Setup

SWMM version 5.0.022 was used to model Rocky Branch Watershed and simulate the effects of adding rain gardens at the parcel-scale to reduce peak storm flows in the main branch. SWMM is a dynamic, open-source computer model that tracks the quantity and quality of the runoff in urban watersheds for either single-event or continuous simulations (Rossman, 2012).

SWMM routes runoff from subcatchments through a network system consisting of pipes, channels, storage/treatment devices, pumps, and regulators. SWMM can model the hydrological performance of typical LID controls such as bioretention cells (or rain gardens), infiltration trenches, porous pavement, rain barrels, and vegetative swales. These LID techniques can be placed within the desired subcatchments at any size and spatial coverage (Qin et al., 2013). SWMM has been used extensively to evaluate the effects of several conventional drainage systems and LID designs in stormwater management (Abi Aad et al., 2010; Elliott and Trowsdale, 2007; Qin et al., 2013; Zahmatkesh et al., 2015; Zoppou, 2001).

The SWMM model simulates three primary processes: infiltration, surface runoff, and flow routing. The infiltration method used is an approach adopted from National Resource Conservation Service (NRCS) curve number (CN) method for estimating runoff. Manning's equation was used for overland flow. The dynamic wave routing method was used for channel routing because it can account for channel storage, backwater, entrance/exit losses, flow reversal, and pressurized flow. This method solves the one-dimensional Saint Venant flow equations, which consist of continuity and momentum equations for conduits and a volume continuity equation at nodes that allows for representing a full closed-conduit pressurized flow. LID techniques are represented in the model as a combination of vertical layers that have specific properties defined on a perunit area basis. Infiltration rates in bioretention cells, which are also called rain gardens, are simulated by the model. Zhang et al. (2010) conducted field experiments investigating the SWMM representation of bioretention facilities and found that the SWMM's representation matched observed peak flow reduction (77% from the model compared to 82% from observations).

Rain gardens were selected as the LID implementation because they could be adopted widely within the watershed and offered significant storage and volume reduction capacity. The

rain garden properties and characteristics were obtained from three resources: Wisconsin Department of Natural Resources Conservation Practice Standard (Bannerman and Considine, 2003), Maryland 2000 Stormwater Management Design Manual (MDE, 2000; Schueler and Claytor, 2000), and Delaware Green Technologies Design Manual and Model (DNREC, 2005). Lucas (2005) includes easy-to-follow guidance for siting, sizing, installing, and planting a rain garden. A rain garden consists of three layers: surface, soil, and storage (Table 2.1). In the model, the total depth used for the soil and storage layers is 120 cm, which is the maximum recommended depth, while the surface layer storage depth (ponding depth or berm height) is varied between the minimum and maximum recommended depths of 10 cm and 30 cm.

Table 2.1 Specifications and characteristics of rain gardens to be implemented in Rocky Branch Watershed.

Layer/Parameter	Value
Surface	
Storage depth (mm)	100 - 300
Vegetation (volume fraction)	0.5
Surface roughness	0
Surface slope	0
Soil	
Soil thickness (mm)	900
Porosity	0.44
Field capacity	0.15
Wilting point (volume fraction)	0.1
Conductivity (mm/hr)	30
Conductivity slope	10
Suction head (mm)	60
Storage	
Storage height (mm)	300
Storage void ratio	0.75
Storage conductivity (mm/hr)	250

2.3.2 Data Preparation

The Rocky Branch Watershed was delineated into 134 subcatchments using a LiDAR-derived 3 m resolution digital elevation model (DEM) (Figure 2.2). The discretization was done by placing subcatchment outlets at 50 m intervals along natural (irregular) portions of the stream channel and at 30 m intervals along the concrete-lined and conduit portions of the stream channel. Stormwater inlets along the streamline were also designated as subcatchment outlets. Standard Geographical Information System (GIS) procedures were used to delineate the subcatchment boundaries. The delineated subcatchments were verified by available orthoimagery data.

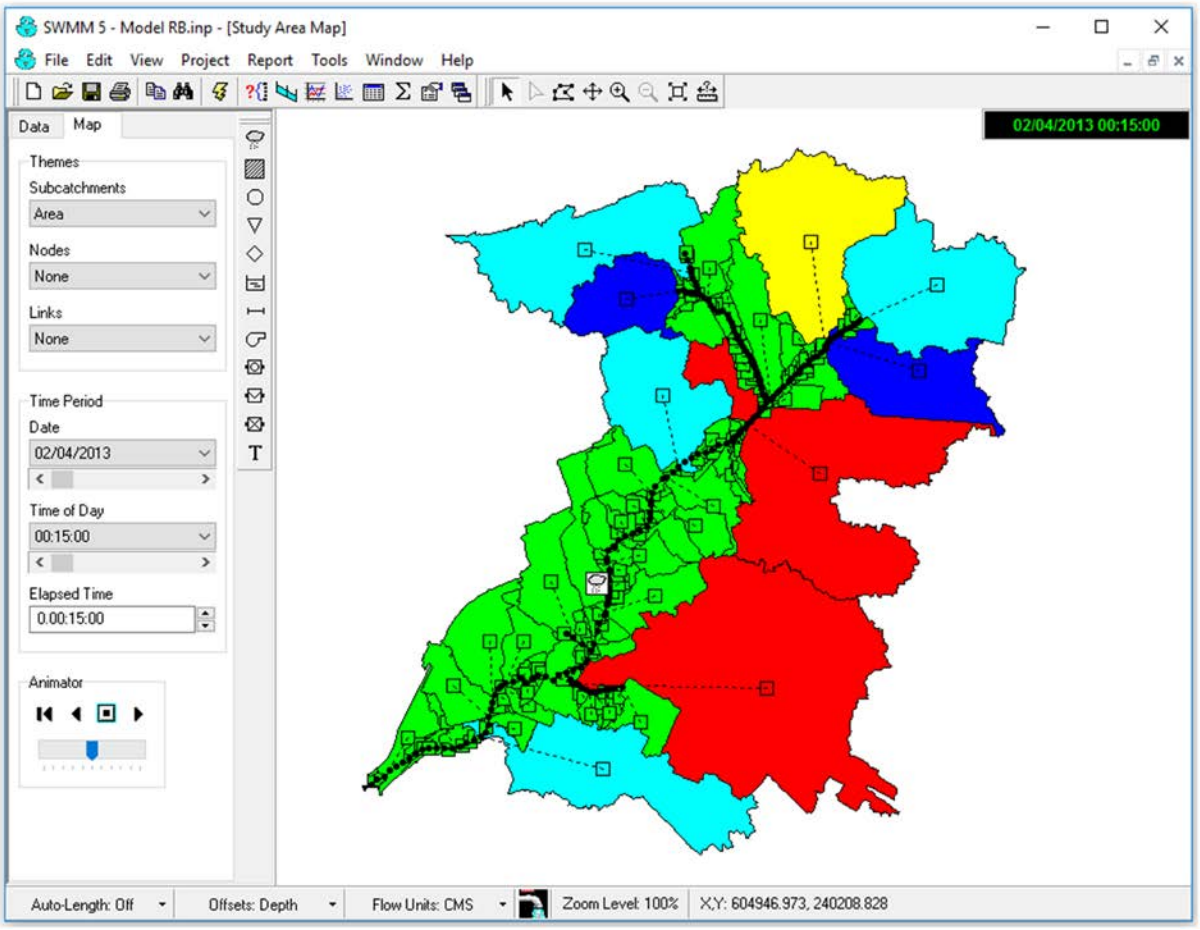


Figure 2.2 Depiction of Rocky Branch Watershed in SWMM model with 134 subcatchments and 188 conduits.

Rocky Branch consists of natural (irregular) channels, pipe sections, and concrete-lined channels that were represented in the model as 188 conduits (Figure 2.3). The Manning's roughness coefficient for the natural cross sections was assumed to be between 0.03 and 0.04, while the Manning's roughness coefficient of the concrete lining cross sections was assumed to be between 0.011 and 0.015. The pipe sections included circular, box, and arch cross sections. Each of the 188 conduits in the model was assigned a cross sectional profile. The cross-sectional profiles were obtained from a combination of LiDAR and ground survey data, and a sample of the profiles was verified by site visits. Figure 2.3 shows example cross sections as they appear within the model for the two locations along the branch where there are stream gauges.

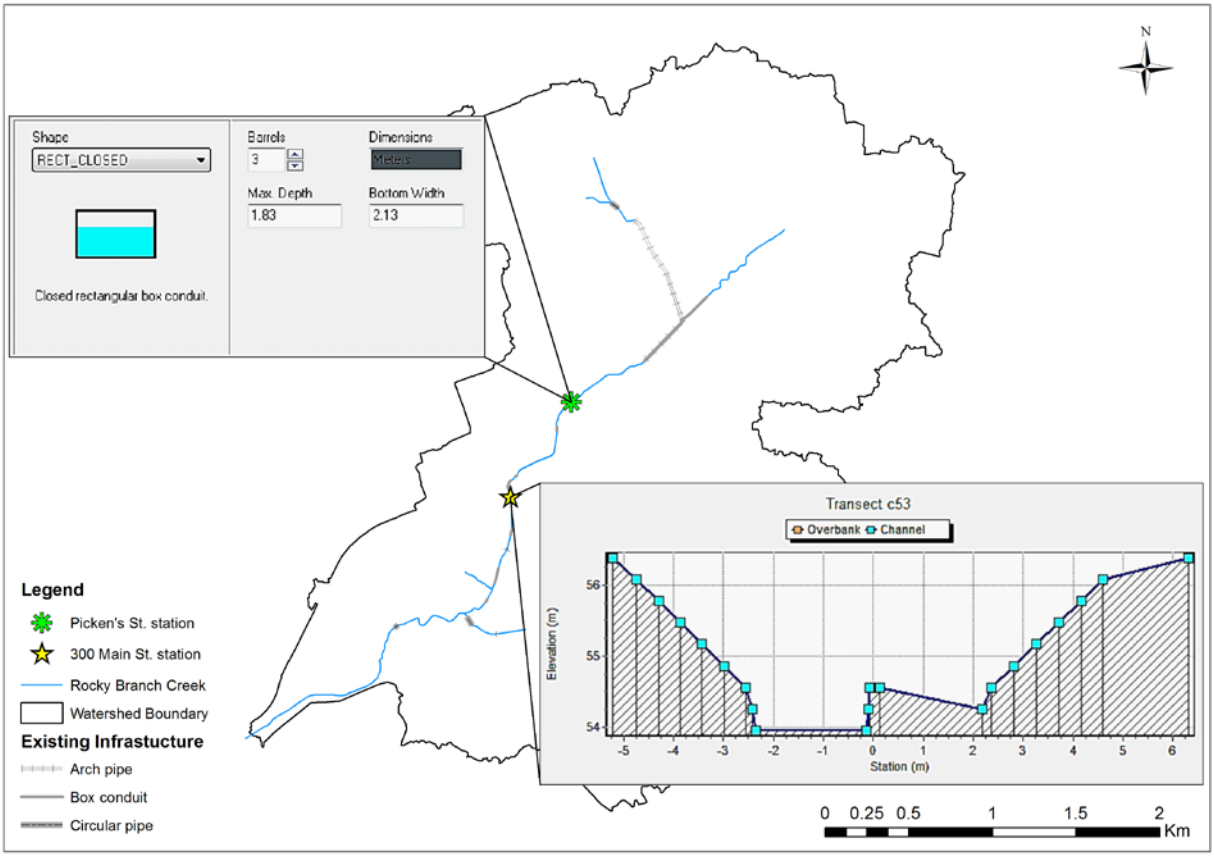


Figure 2.3 Rocky Branch showing stormwater infrastructure lines and cross-sectional types; example cross sections are shown for the two stream gauge locations.

Land use, soils, slope, and CN values were derived using publically available geospatial data sets and GIS processing. Land-use and imperviousness data sets were obtained from the NLCD 2006 (Fry et al., 2011), the latest available version at the time of the initial model development activities. The 2011 and 2006 NLCD data were compared, and no significant differences were found for the study area. NLCDs are raster data, where each pixel is 30 m x 30 m. The Soil Survey Geographic (SSURGO) data sets were downloaded from the USDA for Richland County, South Carolina (SSURGO, 2012). The SSURGO data set is a vector polygon data set with attributes describing soil properties, including Soil Hydrologic Groups. According to these data, 72% of the watershed area is Group B soils, 22% Group A, and 6% Group C. The land use and soil data were used to derive CN values for each subcatchment using NRCS values (Cronshey, 1986) and the processing steps shown in Figure 2.4. Finally, average slopes for each subcatchment were obtained from the DEM used for watershed delineation.

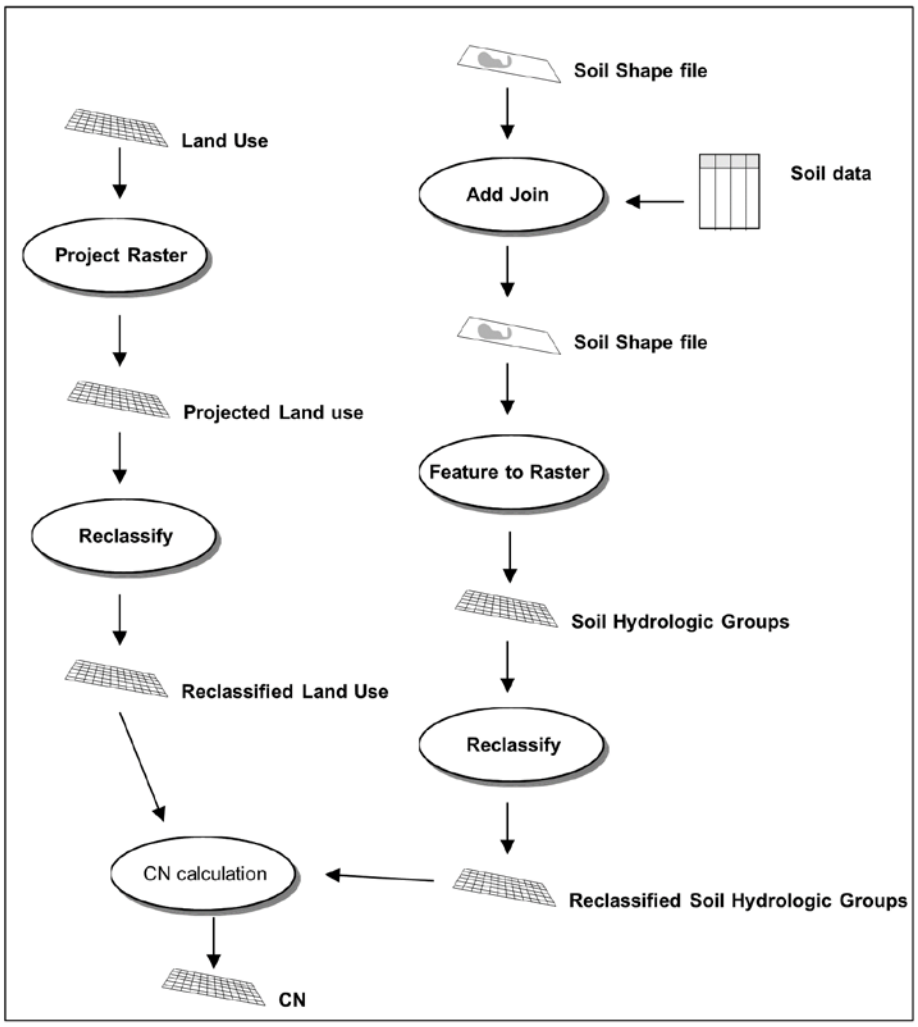


Figure 2.4 GIS workflow for CN computation using land use and soil hydrologic group datasets.

2.3.3 Observed Storm Events

Rainfall data were collected during the study period of June 2012 - June 2013 using a tipping bucket gauge (TR-525USW) (Figure 2.5-a) located at the University of South Carolina's 300 Main Street engineering building (Figure 2.1-a). The factory calibration of the gauge is 0.254 mm (0.01 in.) per tip. The gauge was connected to an electronic data logger (Sutron 8210 A Data Collection Platform). The gauge was installed in a clear and unobstructed mounting location. The stage was measured at the 300 Main Street station for the study period using a bubbler water level gauge (Sutron 8210 A) (Figure 2.5-b). Stage and streamflow data were also obtained from the

USGS (Station 02169505) for the Pickens Street station. These streamflow data at this USGS station were obtained using an Acoustic Doppler Current Profiler (ADCP) (Levesque and Oberg 2012). Figures 2.6 and 2.7 show the observed data for two rainfall events at the two stations: the February 7, 2013, and July 10, 2012, storms, respectively. These storms are presented as representative examples of all six storms. The February 7, 2013, storm was used for model calibration, and the model results for this storm were typical of the other two storms used for calibration. The July 10, 2012, storm was the largest observed storm event and was used for model evaluation. Baseflow at the start of the storm events, which is typically very low, was subtracted from the hydrographs for easier comparison between storm events.

(a) (b)

Figure 2.5 (a) The rain gauge and (b) the stage gauge used to collect observation data at 300 Main Street.

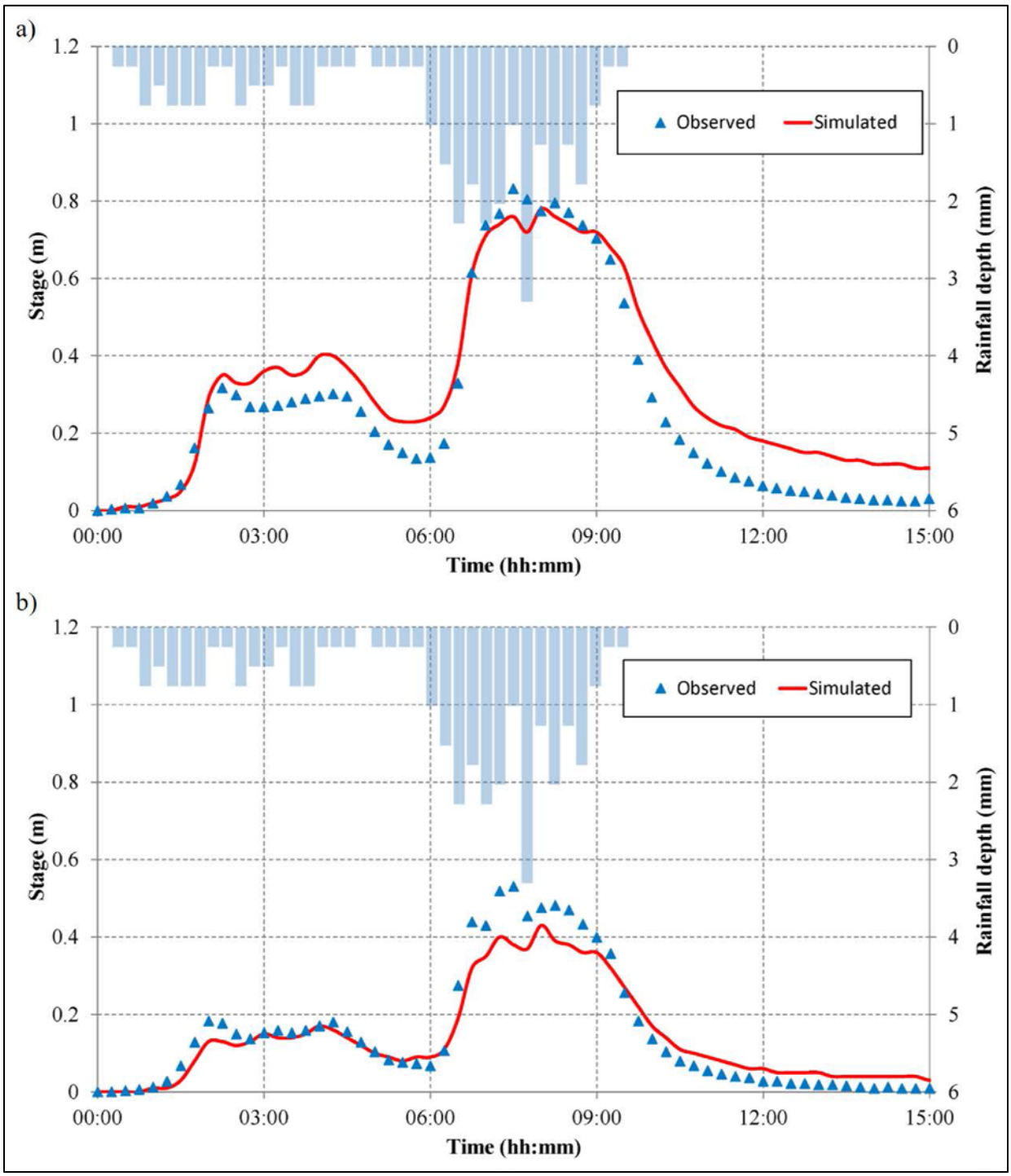


Figure 2.6 Rainfall intensity and corresponding observed and simulated stage for the February 7, 2013 event at two locations along the river channel: (a) 300 Main Street; (b) Pickens Street; this event, which did not cause flooding, was used for model calibration.

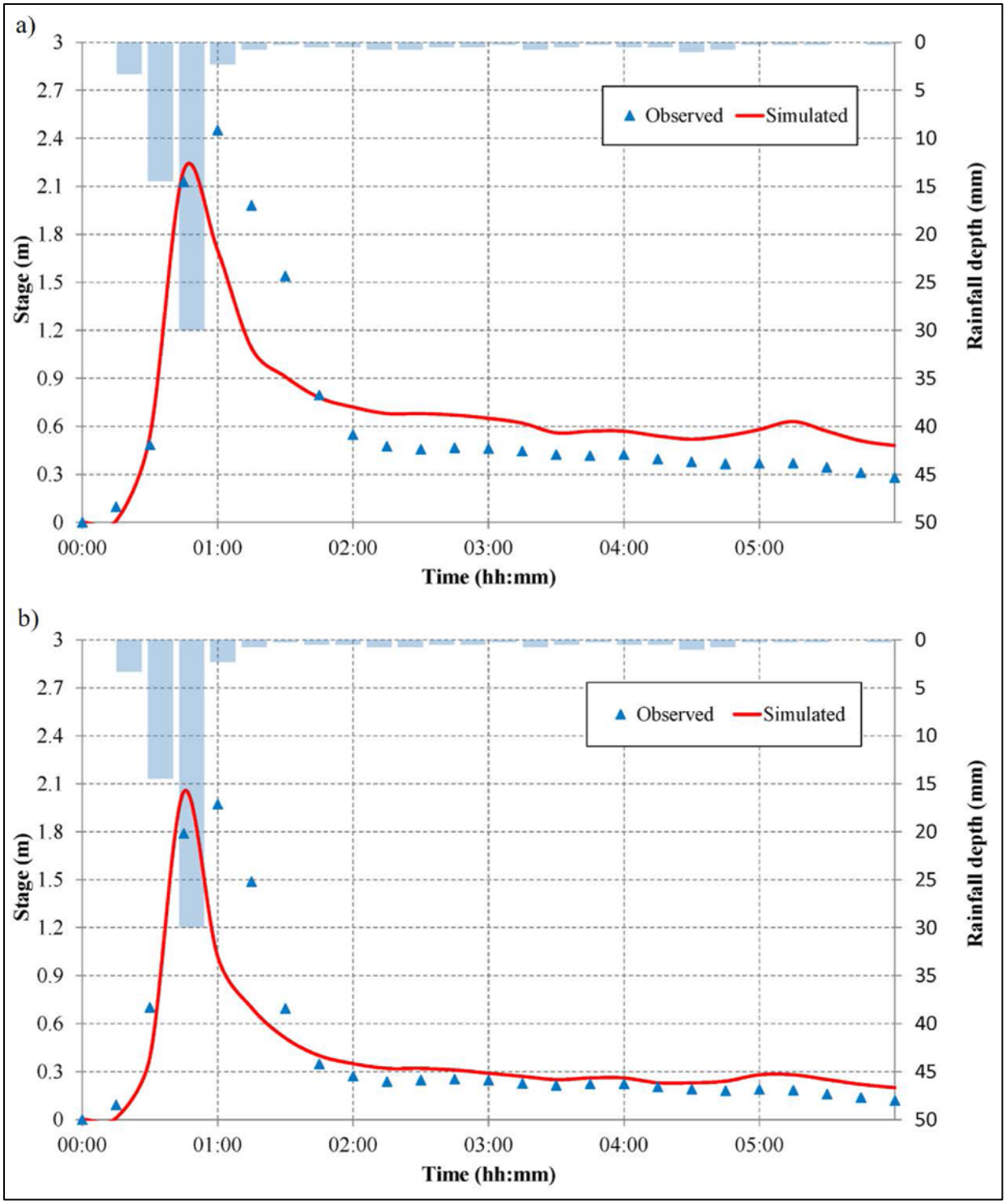


Figure 2.7 Rainfall intensity and corresponding observed and simulated stage for the July 10, 2012 event at two locations along the river channel: (a) 300 Main Street; (b) Pickens Street; the event was used for model evaluation; this was the largest of the observed storm events and caused flooding within the watershed.

2.3.4 Model Calibration and Evaluation

Model calibration was conducted using the subcatchment CN values as the calibration parameter. Each subcatchment in the study area was assigned a specific CN value. The CN values were uniformly changed by a percentage increase or decrease while also ensuring the adjusted values were within the 25 - 98 range. The objective of the calibration was to minimize the relative error between observed and modeled stream stage and flow peaks available at the two gauging locations. This was done for three of the six observed storm events (Table 2.2) using a manual calibration procedure that prioritized accuracy for the large storm event on July 11, 2012, that caused flooding. The three storms selected for calibration were chosen to cover different rainfall depths, durations, and seasons. The calibrated model was evaluated by comparing predicted and observed peak flow and stage values for the remaining three observed storm events. The results of the model calibration and evaluation are described in the "Results and Discussion" section.

Table 2.2 Properties of observed storm events used for model calibration and evaluation.

Storm date	Duration (hh:mm)	Cumulative rainfall depth (mm)
July 11, 2012	00:55	35.05
September 4, 2012	04:23	10.41
February 7, 2013	09:40	33.02
July 10, 2012	01:02	50.29
August 20, 2012	06:33	16.26
September 18, 2012	01:09	7.37

2.3.5 Model Scenarios

Three model scenarios were conducted to address the research objective of improving understanding of how the rate of adoption of LID practices at the parcel level in an urbanized watershed impacts runoff detention. This was done by modeling the peak stage while varying three

key model variables: ponding depth, rain garden area, and diverted runoff. Ponding depth is the maximum depth water can pond in the rain garden before overflowing (i.e., the rain garden berm height). Rain garden area is the total rain garden area as a percentage of the watershed impervious area. Together, these two variables control each rain garden's storage potential. Diverted runoff is the percentage of the runoff generated from impervious surfaces that is diverted to rain gardens. Three general scenarios were investigated in the study.

Scenario 1: Assume all runoff generated on impervious surfaces is diverted to a rain garden (best-case scenario). Introduce rain gardens with total area equal to 10% and then 20% of the impervious surface area in the watershed. Model the reduction in stage at the 300 Main Street station for the event on July 10, 2012.

Scenario 2: Let the ponding depth vary between 20 cm and 30 cm and the rain garden area vary from 15% to 30% of the impervious surfaces in the watershed. Use the model to determine the fraction of runoff from impervious surfaces that must be diverted to rain gardens to reduce the peak stage at the 300 Main Street station below bankfull stage for different combinations of ponding depth and rain garden area.

Scenario 3: Fix the ponding depth at 30 cm for maximum storage potential. Increase the storm size to 5, 10, 25, and 50 year return period storms. Use the model to determine the rain garden area and diverted runoff required to reduce the peak stage at the 300 Main Street station below bankfull conditions for these larger storm events.

According to the Precipitation Frequency Data Server (PFDS) and using the COLUMBIA UNIV OF SC, 38-1944 station, the July 10, 2012, storm was equivalent to a 2 year return period, 1 hour event (Bonnin et al., 2006). Information from PFDS was used to generate storms with larger return periods while maintaining the rainfall pattern and 1 hour storm duration from the July 10,

2012, event. This was done by normalizing the July 10, 2012 event and then multiplying this normalized storm by the total precipitation depth for a 5, 10, 25, and 50 year return period, 1 hour duration event obtained from PFDS. Thus, these are synthetic storms with higher return periods but the same rainfall pattern and duration as the storm observed on July 10, 2012.

2.4 Results and Discussion

2.4.1 Model Calibration and Evaluation

The final calibration resulted in the CN values for all subwatersheds being reduced by 15% from their initial value uniformly throughout the watershed. This calibration resulted in the July 11, 2012, event having the lowest error with the two stage depth predictions within 1% relative error and the discharge within 8% relative error (Table 2.3). The relative errors were greater for the other two storms used in the calibration stage. The February 7, 2013, event (Figure 2.6) was generally between the other two storms in terms of relative error for predicting peak flows. The highest relative errors were for the September 4, 2012, event at the Pickens Street monitoring station, with relative errors of 30 - 35% for both stage and flow. For the same event, the stage relative error was only 8% for the 300 Main Street station. The September 4, 2012, event was a relatively minor event, so this was deemed an acceptable error given that the primary objective of the calibration was to match the larger July 11, 2012, storm event that resulted in flooding.

Three independent storm events not considered when calibrating the model were used for evaluating the model. Results of the model evaluation show that the relative errors of both stage and discharge were less than 12% for two of the three storm events (see Table 2.3 for statistics for all three storms and Figure 2.7 for the simulated hydrograph for the storm event that occurred on July 10, 2012). The August 20, 2012, storm had higher relative errors, showing that the model

under predicted both stage and flow at both observation stations for this event. A likely explanation for this is differences in the antecedent moisture condition (AMC) between the August 20, 2012, storm and the other two storm events. The model assumes normal AMC (AMC II), which is consistent with the two storms that had lower relative errors. Rainfall records for the 5 days prior to the August 20, 2012, event, however, suggest wet AMC (AMC III). To account for this, CN parameters in the model for the August 20, 2012, storm could be adjusted to account for wet AMC, and this would reduce the relative error because it would increase the amount of runoff predicted by the model. Given that the model scenarios performed in this study assume normal AMC, the model was considered acceptable for the purposes of the study.

Table 2.3 Relative error between modeled and observed channel stage and flow at two stations for storm events used in model calibration and evaluation.

Storm events	Calibration			Evaluation		
	July 11, 2012	**September 4, 2012**	**February 7, 2013**	**July 10, 2012**	**August 20, 2012**	**September 18, 2012**
300 Main St. Station						
Stage (m)						
Observed	1.63	0.70	0.84	2.45	0.79	0.66
Modeled	1.62	0.76	0.78	2.15	0.58	0.59
Relative error (%)	-0.65	8.88	-7.58	-12.17	-26.81	-10.14
Pickens St. Station						
Stage (m)						
Observed	1.33	0.38	0.53	1.97	0.50	0.27
Modeled	1.34	0.51	0.46	2.05	0.27	0.27
Relative error (%)	1.06	33.60	-12.66	4.02	-45.73	-2.22
Flow (m3/s)						
Observed	22.80	3.05	5.00	-	4.65	2.80
Modeled	24.61	4.13	5.46	38.35	3.05	2.76
Relative error (%)	7.95	35.36	9.27	-	-34.46	-1.31

2.4.2 Impact of Rain Garden Area on Flood Mitigation

Figure 2.8 shows results from the first model scenario. The channel stage for the July 10, 2012, event at the Main Street monitoring station was reduced to below bankfull conditions through the introduction of rain gardens into the watershed. This is a best-case scenario in that all runoff from impervious surfaces can be diverted to the rain gardens. The cases where rain garden area equals 10% and 20% of the watershed impervious area are presented. The results suggest that including rain gardens with a total area just above 10% of the impervious area within the watershed would reduce the peak stage to below bankfull conditions. If the rain garden area is increased to 20% of the impervious area within the watershed, the flood peak would be further reduced to approximately 0.2 m below the bankfull stage at the 300 Main Street station. Prior studies and guidelines focusing on the water quality and groundwater recharge benefits of rain gardens recommend rain garden areas of 10 - 20% of the impervious area within a watershed (Atchison et al., 2006; Dussaillant et al., 2004), which interestingly would also be sufficient for flood control in this scenario.

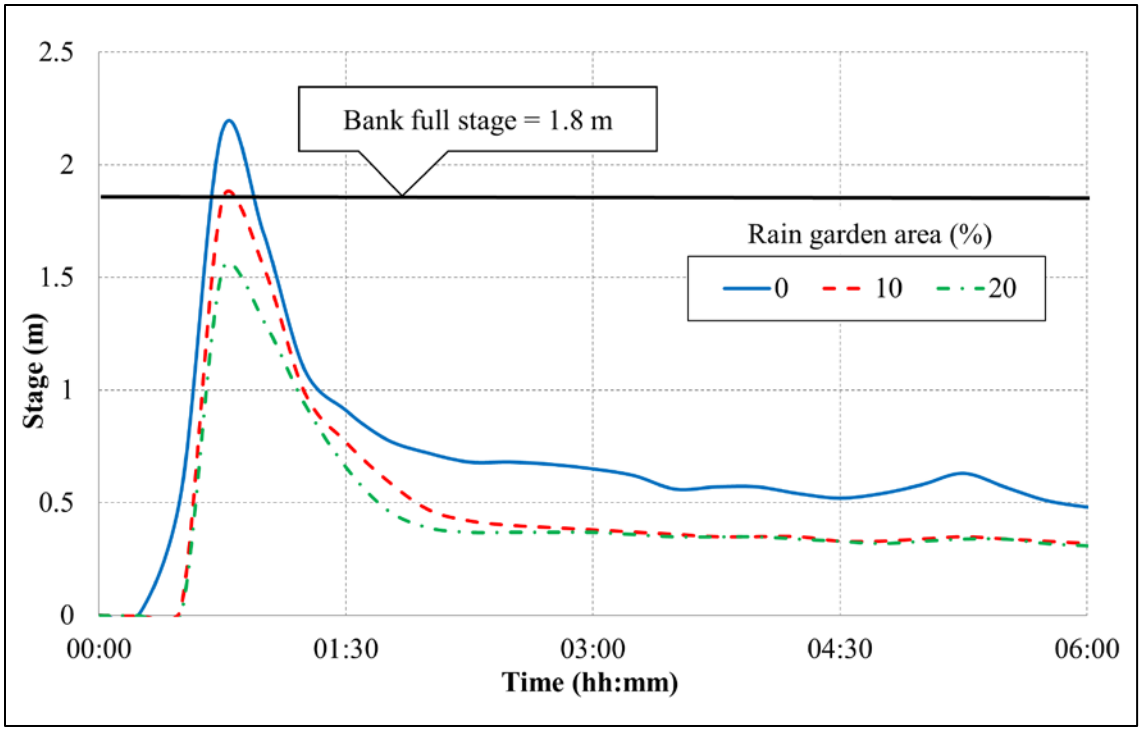


Figure 2.8 Simulated reduction in channel stage for July 10, 2012, storm event at 300 Main Street through introduction of rain gardens into watershed.

2.4.3 Runoff Contribution to Rain Gardens for Flood Mitigation

Results from the second model scenario show the relationship between rain garden area, diverted runoff, and ponding depth for mitigating the July 10, 2012, flood event (Figure 2.9). As an example, consider the case from the prior analysis where ponding depth was equal to 10 cm and the rain garden area was equal to 10% of impervious surfaces. Figure 2.9 shows that 100% of the runoff from an impervious surface would need to be diverted to the rain gardens to not exceed bankfull conditions (i.e., to mitigate flooding). When the rain garden area is increased to 20%, the diverted runoff required to not exceed bankfull conditions decreases significantly to only 20%. This result shows the importance of sufficient rain garden storage to capture excess runoff volume. Figure 2.9 also shows that, once sufficient storage is achieved, either from increasing the ponding depth or increasing the rain garden area, the diverted runoff needed to not exceed bankfull

conditions approaches approximately 15% for this storm event. Finally, Figure 2.9 shows that if a ponding depth of 30 cm is used, the required storage volume is achieved with a rain garden area of 20% and additional rain garden area does not significantly aid flood mitigation for this storm.

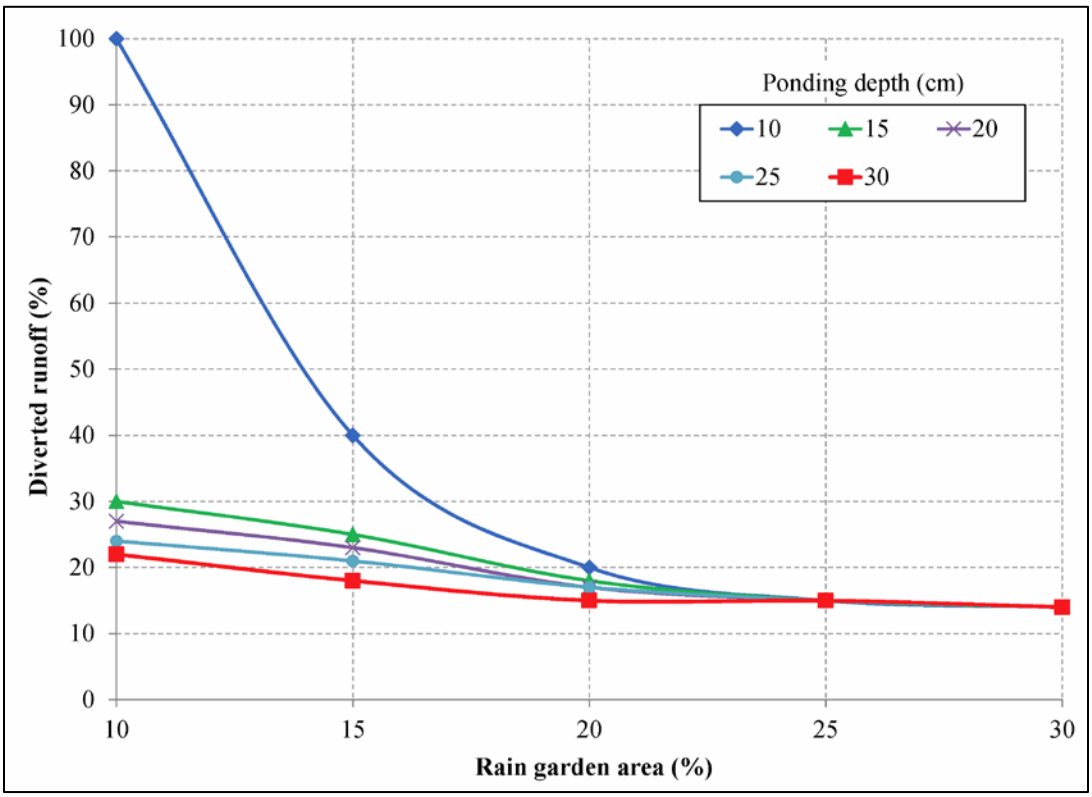


Figure 2.9 Required runoff diversion to mitigate flooding from July 10, 2012, event as function of rain garden area and ponding depth.

2.4.4 Impact of Storm Return Period on Flood Mitigation

Results from the third model scenario show how rain gardens could mitigate flooding for larger storm events. The prior scenarios focused on the July 10, 2012, event, which was determined to be a 2 year return period, 1 hour duration event. In this scenario, synthetic storms with higher return periods are used in the model (Figure 2.10). The hyetographs for these synthetic storms were generated using the same rainfall pattern and duration of the July 10, 2012, event, but with a total rainfall depth consistent with larger storm events as described in the "Materials and Methods"

section. Using these hyetographs and assuming a 30 cm rain garden ponding depth for maximum storage, Figure 2.11 shows the relationship between rain garden area and diverted runoff required for flood mitigation. Again this result shows a steep curve when rain garden storage is limited. Each return period approaches a diverted runoff value once sufficient volume is achieved. These diverted runoff values represent the runoff reduction required to mitigate flooding for the larger storm events.

Assuming a given rain garden area and ponding depth, it is possible to determine the diverted runoff required to mitigate flooding for different return period storms (Figure 2.12). Figure 2.12 shows results for a rain garden area equal to 20% and 30% of the impervious area. For both cases, ponding depth is set to 30 cm for maximum storage potential. For return period storms less than 10 year, there is little difference between 20% and 30% rain garden areas. This suggests that both scenarios have sufficient storage to mitigate flooding for equivalent storms. Therefore, there is little to be gained from adding rain gardens with a total area exceeding 20% of the watershed's impervious cover for storms with return periods less than or equal to 10 year. For the 10 year return period storm, approximately 38% of the runoff from impervious surfaces should be diverted to rain gardens to mitigate flooding. For the 5 year return period storm, approximately 27% of runoff should be diverted, and this value drops to 15% for the 2 year return period storm. Storms with greater than a 10 year return period require that more than 50% of the runoff in the study watershed be diverted for flood mitigation.

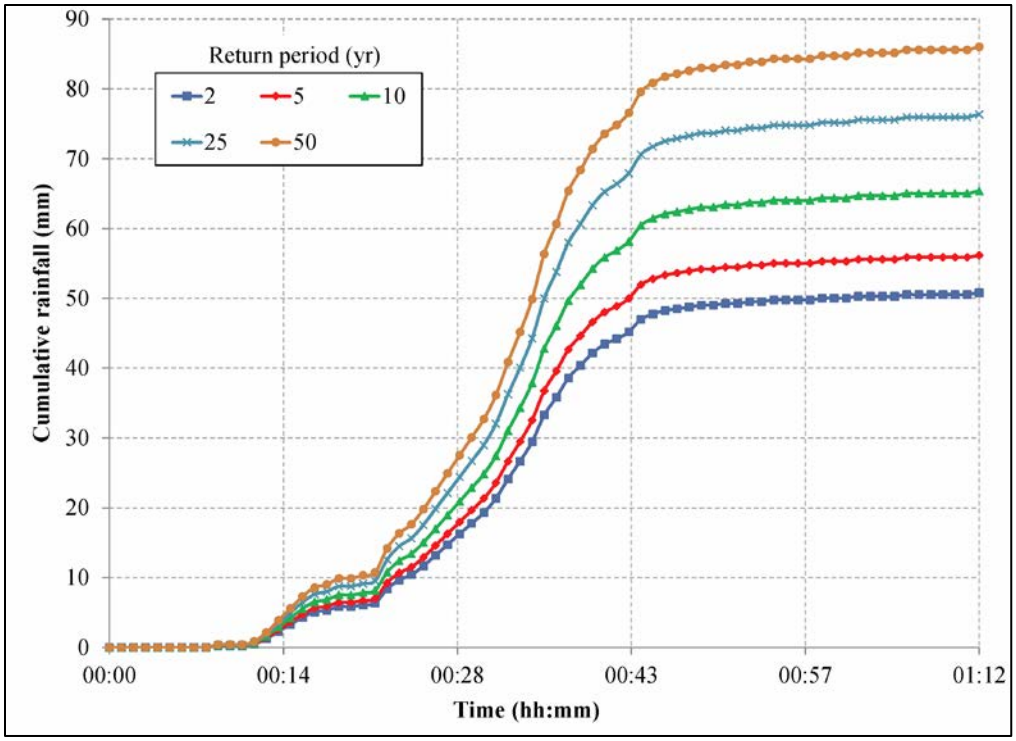


Figure 2.10 Synthetic storm events with higher return periods and 1 hour duration; the rainfall pattern for each synthetic storm is based on observed July 10, 2012, storm event.

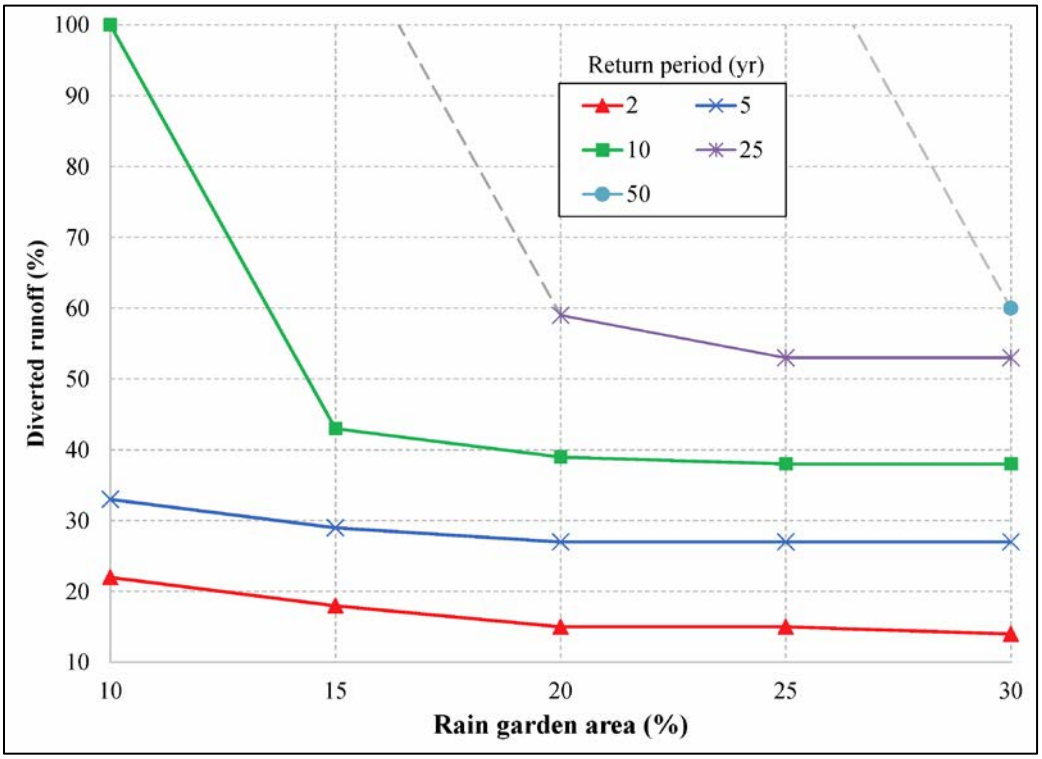


Figure 2.11 Percentage of runoff from impervious surfaces that must be diverted to rain gardens to mitigate flooding as a function of rain garden area (assumes a 30 cm ponding depth for rain gardens).

Given these required rain garden areas and diverted runoff amounts, the question becomes what level of LID adoption is reasonable within the watershed. Bakacs et al. (2013) found that, following an educational training program, 48% of respondents in Virginia and 58% of respondents in New Jersey adopted a stormwater BMP at their homes. The majority of the respondents who took action redirected downspouts to gardens or mulched areas (64% and 54%, respectively). A much smaller fraction of respondents (12% and 4%, respectively) installed a rain garden. Given that 35% of the watershed's impervious cover is rooftop area, redirecting downspouts to existing gardens or mulched areas with sufficient storage to reduce runoff could have a significant impact. Furthermore, efforts by public entities, including the university, to reduce runoff from impervious surfaces using LID techniques could likewise be significant. Thus, while it is difficult to determine what level of runoff reduction through the adoption of LID techniques is possible in the watershed, it seems reasonable to suggest that the adoption of LID approaches could achieve the storage increase and runoff capture required to mitigate flooding up to a 5 year return period storm (rain gardens with total area equal to 20% of impervious surfaces within the watershed, 27% of the runoff generated from impervious surfaces diverted to rain gardens).

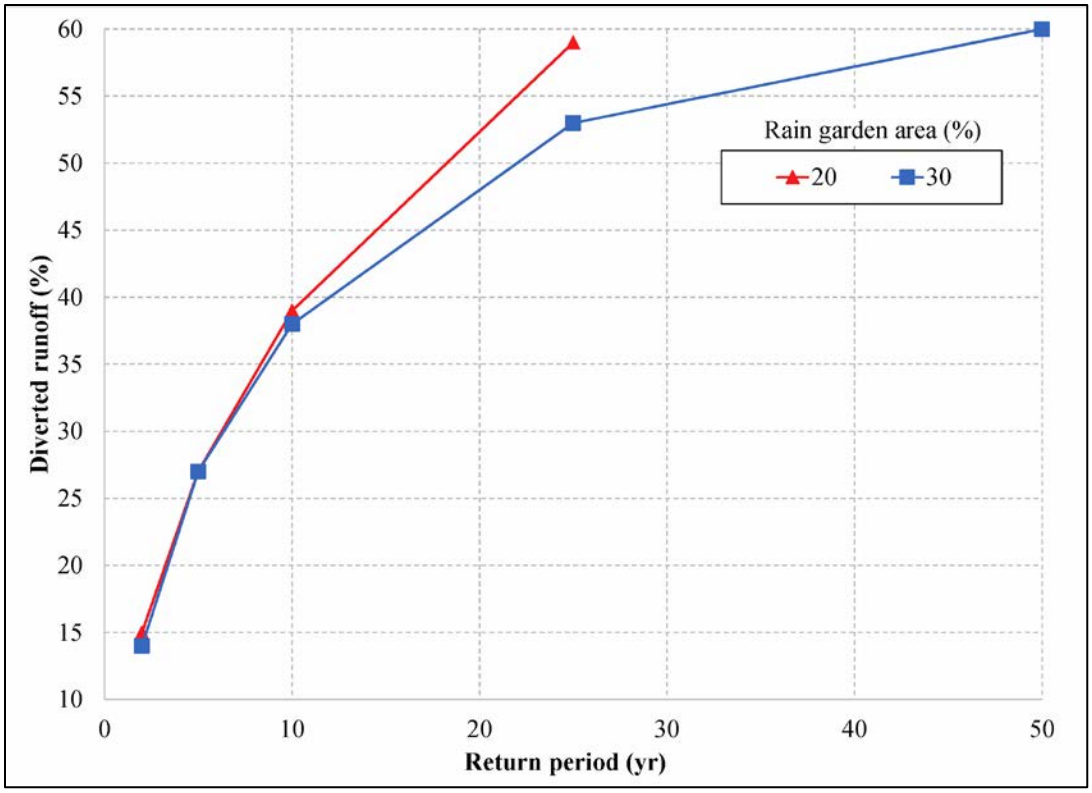


Figure 2.12 Percentage of runoff from impervious surfaces that must be diverted to rain gardens to mitigate flooding as a function of storm return period (assumes a 30 cm ponding depth for rain gardens).

2.5 Conclusions

The primary objective of this study was to improve understanding of how the adoption of LID practices, in particular rain gardens, at the parcel level in an already urbanized watershed might impact runoff detention and, therefore, flood risk. By understanding the required conditions under which distributed storm water controls like rain gardens could mitigate flooding, it is possible to suggest the potential and limitations of the approach. Ultimately stormwater control measures are used in combination to address water quality and quantity issues in developed watersheds, so these modeling scenarios are meant more for providing bounds on LID techniques, and rain gardens in particular, as a flood mitigation strategy.

The first challenge in addressing flooding in an urbanized watershed is to provide sufficient volume for storing runoff generated from impervious surfaces in the watershed. In this study, the storage volume added by the rain gardens was the product of two model variables: the total area of the rain gardens in the watershed as a percentage of the total impervious surface and the ponding depth (or berm height) of the rain gardens. Typical values for rain garden area cited in prior work focusing on water quality and groundwater recharge benefits of rain gardens have been 10 - 20% of the impervious area (Dussaillant et al., 2004). The results suggest that 20% is a sufficient area to mitigate flooding for storm events with less than or equal to a 10 year return period if the maximum recommended ponding depth of 30 cm is used.

Once sufficient storage is available, the next challenge is diverting runoff from impervious surfaces to locations like rain gardens, where it can infiltrate. Using modeling scenarios for the study watershed, it was determined that 15% of runoff from impervious surfaces would need to be diverted to mitigate flooding for a 2 year return period, 1 hour duration storm. For a 5 year return period, 1 hour storm, there would need to be a 27% runoff reduction. Storms with a 10 year return period would require 38% runoff reduction, whereas higher return periods would require over 50% runoff reduction. Given that rooftop areas account for 35% of a watershed's impervious cover, and research suggests approximately 50 - 60% adoption rates of LID techniques by homeowners following an outreach campaign (Bakacs et al., 2013), the results of this study suggest that distributed LID approaches could potentially be used to mitigate up to a 5 year return period storm. However, further research on possible adoption rates within the study watershed is needed to verify this conclusion.

2.6 References

Abi Aad, M. P., Suidan, M. T., Shuster, W. D., 2010. Modeling techniques of best management practices: Rain barrels and rain gardens using EPA SWMM-5. J. Hydrol. Eng., 10.1061/(ASCE)HE.1943 -5584.0000136, 434-443.

Arnold, C. L., Jr., Gibbons, C. J., 1996. Impervious surface coverage: The emergence of a key environmental indicator. J. Am. Plan. Assoc., 62(2), 243-258.

Atchison, D., Potter, K., Severson, L., 2006. Design guidelines for stormwater bioretention facilities accessed August 18, 2015. http://aqua.wisc.edu/publications/PDFs/stormwaterbioretention.pdf.

Bakacs, M. E., et al., 2013. Rain barrels: A catalyst for change. J. Extension, 51(3), 3RIB6.

Bannerman, R. T., Considine, E., 2003. Rain gardens: A how-to manual for homeowners. Wisconsin Dept. of Natural Resources, Madison, WI.

Bedan, E. S., Clausen, J. C., 2009. Stormwater runoff quality and quantity from traditional and low impact development watersheds. J. Am. Water Resour. Assoc., 45(4), 998-1008.

Bonnin, G. M., Martin, D., Lin, B., Parzybok, T., Yekta, M., Riley, D., 2006. Precipitation-frequency atlas of the United States, Volume 2, Version 3: Delaware, District of Columbia, Illinois, Indiana, Kentucky, Maryland, New Jersey, North Carolina, Ohio, Pennsylvania, South Carolina, Tennessee, Virginia,West Virginia. U.S. Dept. of Commerce, Silver Spring, MD.

Cronshey, R., 1986. Urban hydrology for small watersheds, Urban hydrology for small watersheds. U.S. Dept. of Agriculture, Soil Conservation Service, Engineering Division, Washington, DC.

Damodaram, C., et al., 2010. Simulation of combined best management practices and low impact development for sustainable stormwater management. JAWRA J. Am. Water Resour. Assoc., 46(5), 907-918.

Davis, A. P., 2005. Green engineering principles promote low-impact development. Environ. Sci. Technol., 39(16), 338 A-344 A.

Davis, A. P., Hunt, W. F., Traver, R. G., Clar, M., 2009. Bioretention technology: Overview of current practice and future needs. J. Environ. Eng., 10.1061/(ASCE)0733-9372(2009)135:3(109), 109-117.

Dietz, M. E., 2007. Low impact development practices: A review of current research and recommendations for future directions. Water. Air. Soil Pollut., 186(1), 351-363.

DNREC (Delaware Natural Resources and Environmental Control), 2005. Green technology: The Delaware urban runoff management approach. Delaware Dept. of Natural Resources and Environmental Control, Division of Soil and Water Conservation, Dover, DE.

Dussaillant, A. R.,Wu, C. H., Potter, K.W., 2004. Richards equation model of a rain garden. J. Hydrol. Eng., 10.1061/(ASCE)1084-0699 (2004)9:3(219), 219-225.

Elliott, A., Trowsdale, S., 2007. A review of models for low impact urban stormwater drainage. Environ. Model. Software, 22(3), 394-405.

Fry, J. A., et al., 2011. Completion of the 2006 national land cover database for the conterminous United States. Photogramm. Eng. Remote Sens., 77(9), 858-864.

Lee, J. G., Heaney, J. P., 2003. Estimation of urban imperviousness and its impacts on storm water systems. J. Water Resour. Plan. Manage., 10.1061/(ASCE)0733-9496(2003)129:5(419), 419-426.

Levesque, V. A., Oberg, K. A., 2012. Computing discharge using the index velocity method accessed August 18, 2015. http//pubs.usgs.gov/tm/3a23.

Line, D. E., Brown, R. A., Hunt,W. F., Lord,W. G., 2011. Effectiveness of LID for commercial development in North Carolina. J. Environ. Eng., 10.1061/(ASCE)EE.1943-7870.0000515, 680-688.

Loperfido, J. V., Noe, G. B., Jarnagin, S. T., Hogan, D. M., 2014. Effects of distributed and centralized stormwater best management practices and land cover on urban stream hydrology at the catchment scale. J. Hydrol., 519, 2584-2595.

Lucas, W. C., 2005. Green technology: The Delaware urban runoff management approach. A technical manual for designing nonstructural BMPs to minimize stormwater impacts from land development. Delaware DNREC, Dover, DE.

MDE (Maryland Department of the Environment)., 2000. 2000 Maryland stormwater design manual, Vols. I and II. Center for Watershed Protection and the Maryland Dept. of the Environment, Water Management Administration, Baltimore.

Monk, J., Holleman, J., 2010. Heavy rains flood parts of Columbia accessed August 18, 2015. http://www.thestate.com/news/local/article14387762.

NOAA, 2010. Five points flooding accessed August 18, 2015. http://www.erh.noaa.gov/cae/Events/FivePointsFlood.

Page, J. L.,Winston, R. J., Mayes, D. B., Perrin, C., Hunt,W. F., 2015. Retrofitting with innovative stormwater control measures: Hydrologic mitigation of impervious cover in the municipal right-of-way. J. Hydrol., 527, 923-932.

Qin, H., Li, Z., Fu, G., 2013. The effects of low impact development on urban flooding under different rainfall characteristics. J. Environ. Manage., 129, 577-585.

Roesner, L. A., Urbona, B. R., 1998. Urban runoff quality management. ASCE, Reston, VA.

Rossman, L., 2012. Storm Water Management Model (SWMM) - User's manual, version 5.0 accessed August 18, 2015. http//www.epa.gov/nrmrl/wswrd/wq/models/swmm.

Santaella, T., Gillbert, J., 2011. Five points businesses recover from flood waters accessed August 18, 2015. http://archive.wltx.com/news/article/152992/2/Five.

Schueler, T. R., 1995. Site planning for urban stream protection. Metropolitan Washington Council of Governments, Washington, DC.

Schueler, T. R., Claytor, R. A., 2000. Maryland stormwater design manual. Maryland Dept. of the Environment, Baltimore.

Selbig,W. R., Bannerman, R. T., 2008. A comparison of runoff quantity and quality from two small basins undergoing implementation of conventional and low-impact-development (LID) strategies: Cross plains, Wisconsin, water years 1999 - 2005. USGS, VA.

Shuster, W. D., Rhea, L., 2013. Catchment-scale hydrologic implications of parcel-level stormwater management (Ohio USA). J. Hydrol., 485, 177-187.

SSURGO (Soil Survey Geographic), 2012. Soil survey staff, natural resources conservation service accessed November 10, 2012. http://sdmdataaccess.nrcs.usda.gov/.

The State, 2012. Severe weather pummels the Midlands accessed August 18, 2015. http://www.thestate.com/news/local/article14406569.

The State, 2014. Heavy rain causes flash flooding in Midlands, severe flooding in upstate accessed August 18, 2015. http://www.thestate.com/news/local/article13875068.

WIS TV, 2015. Morning rain after another evening of damaging, flooding storms accessed August 18, 2015. http://www.wistv.com/story/29724275/severe-storms.

Zahmatkesh, Z., Burian, S. J., Karamouz, M., Tavakol-Davani, H., Goharian, E., 2015. Low-impact development practices to mitigate climate change effects on urban stormwater runoff: Case study of New York City. J. Irrig. Drain. Eng., 10.1061/(ASCE)IR.1943-4774 .0000770, 04014043.

Zhang, G., Hamlett, J. M., Saravanapavan, T., 2010. Representation of low impact development scenarios in SWMM. Dyn. Model. Urban Water Syst. Monogr., 18, 12.

Zoppou, C., 2001. Review of urban storm water models. Environ. Model. Software, 16(3), 195-231.

Chapter 3: A Cloud-Based Decision Support System for Managing Flooding Impacts to Transportation Infrastructure in Coastal Virginia[2]

3.1 Introduction

Floods were the number one natural disaster in the US in terms of the lives lost and property damage during the 20th century (Perry, 2000). Statistics show that from 2006 to 2015, total flood insurance claims averaged more than $1.9 billion per year (NFIP Statistics, 2016). Rainfall events are predicted to become more frequent and intense due to climate change, which is expected to cause increased flooding (Melillo et al., 2014). As society faces flooding events with increasing frequency and intensity, flood modeling will become an even more important tool for decision makers. Such models can be used to warn municipalities and communities of forecasted flooding impacts. They can also be used to test alternative flood mitigation strategies for addressing flood problems.

The National Research Council (NRC) has recommended increased use of two dimensional (2D) hydrodynamic models for flood risk management purposes (NRC, 2009). There are several advantages to using 2D models over one dimensional (1D) models that include better resolution of velocity, localized depth and surface water elevation, and determination of floodplain extent directly. 2D hydrodynamic models are especially important for cases with complex flows such as in low-relief terrains with flat and/or mild slopes. For these low-relief terrains, 1D models are not sufficient due to the limitations of assumed uniform water velocity and constant water surface elevation modeled on each cross section (Garcia et al., 2015).

[2]This Chapter is in preparation for submission to a peer reviewed journal. The tentative title, authors, and journal for the submission follow:
Morsy, M.M., Goodall, J.L., O'Neil, G., Sadler, J., Voce, Daniel, Hassan, G., Huxley, C. A Cloud-Based Decision Support System for Real-time Warning of Flooding Impacts to Transportation Infrastructure in Coastal Virginia. In preparation for submission to Environmental Modelling & Software.

Executing 2D hydrodynamic models at the regional scale (~$10x10^3$ - $100x10^3$ km^2) requires parallel computation in order to run in a time frame reasonable for flood warning applications. graphical processing units (GPUs) have recently been shown to be an effective way for parallel execution of 2D hydrodynamic models with speed-ups of 20x to 100x (Huxley and Syme, 2016; Garcia et al., 2015; Vacondio et al., 2014; Kalyanapu et al., 2011). Vacondio et al. (2014) expects that GPUs will continue to be attractive for 2D numerical models compared to clusters of central processing units (CPUs) due to (i) fast-developing GPU hardware, (ii) quickly decreasing costs, and (iii) less maintenance compared to large CPU clusters. With the speed-ups provided by GPUs, regional flood warning systems can now be implemented with 2D hydrodynamic models and the spatial resolution needed to provide targeted and detailed information to decision makers.

There are several related efforts aimed at improving flood warnings. The National Weather Service (NWS) and the United States Geological Survey (USGS) have a joint project to generate flood inundation maps at locations where a NWS forecast point and a USGS stream gauge exist (Fowler, 2016). At these locations, a flood inundation map is created for multiple possible water surface elevations and, by using a rating curve and forecasted discharge, the data is converted into the corresponding water surface elevation. Then the corresponding flood inundation map is selected from a precomputed library of flood inundation maps in the United States. The National Flood Interoperability Experiment (NFIE) is a multiagency effort in collaboration with the academic community to improve river and flood forecasts (Maidment, 2016). A key component of NFIE is a model called Routing Application for Parallel computing of Discharge (RAPID) (http://rapid-hub.org/) that was developed to operate on the 2.67 million NHDPlus catchments and uses parallel computing to solve the 1D Muskingum flow equations on this large river network (Maidment, 2016; David et al., 2013, 2011). NFIE showed it was possible to improve the spatial

density of forecast flooding locations by more than 700 times compared with the present NWS river forecast system (Maidment, 2016). However, in some instances a 2D flood model will be necessary to accurately model water transport over large flat areas. Delft-FEWS is a hydrological forecasting and warning framework that provides a platform through which operational forecasting systems can be constructed, allowing for flexibility in the integration of models and data (Werner et al., 2013). Delft-FEWS does not contain any inherent hydrologic model capabilities within its code base. Instead, it relies on the integration of external hydrologic model components.

The objective of this research is to design and prototype a cloud-based system for supporting decision makers as they assess flood risk to transportation infrastructure during extreme weather events. The system automates forecast data access and pre-processing, execution of a high-resolution 2D hydrodynamic model, and map-based visualization of model outputs. This work advances the prior approaches described earlier by presenting a cloud-based framework for modeling regions with complex flows using a 2D hydrodynamic model. Rather than relying on precomputed flood maps, flood depths and extents, this approach allows for modeling water flows in real-time based on current and forecasted conditions. It is an approach that could be adopted in Delft-FEWS to leverage cloud and GPU resources within this general framework.

The overall goal of this study is to design and implement an automated flood warning system using a sophisticated 2D hydrodynamic model and modern cloud-based cyberinfrastructure and computing resources. The study advances on previous work funded by the Virginia Department of Transportation for the Hampton Roads District of Virginia that produced the Regional River Severe Storm Model (R^2S^2) (Hassan Water Resources PLC., 2012). The purpose of R^2S^2 is to help Residency Administrators to efficiently allocate scarce resources to close roads and to assist first responders with entering and exiting flood prone areas. This research advances

R^2S^2 by automating what was previously a manual process of converting forecast rainfall data into model inputs, running the model, and visualizing the results. Furthermore, this research addresses computational challenges with using R^2S^2 for real-time flood warning and emergency management applications. This research also moves R^2S^2 to the cloud and is one of the first cloud-based flood warning applications with (i) an automated workflow for obtaining the real-time forecast rainfall data, (ii) execution of a model to identify flooded bridge and culvert locations in a time duration sufficient for warning and emergency management purposes, and (iii) generation of an online map with locations of the flooded roadways and bridges, and the ability to send automated warning messages via email. This system can provide the Virginia Department of Transportation (VDOT) with information needed when determining road closures, disseminating warning messages for area residents, and making other emergency management decisions that affect human safety and property damage. Although the current system is focused on VDOT as the primary user, it could be expanded in the future to disseminate more general flooding information to other stakeholders.

Cloud computing is gaining attention in environmental applications to satisfy the peak performance needs of applications that use large amounts of processing power (Granell et al., 2016). Sun (2013) used Google Drive, a cloud computing service, to host an environmental decision support systems (EDSS) module that is migrated from the traditional client-server-based architecture. Using Google Drive with the capability of providing a number of basic visual analytics features, the collaboration between the decision makers can be increased and the cost of small scale EDSS can be decreased. Ercan et al. (2014) used the Windows Azure Cloud environment to run a created calibration tool built with the modified calibration method, a parallel version of the Dynamically Dimensioned Search (DDS) for calibrating the Soil and Water Assessment Tool (SWAT) model in Azure. Using this tool, the result showed a significant speed-

up of the model calibration for six different model scenarios. Wan et al. (2014) introduced a public cloud-based flood cyber-infrastructure (CyberFlood). CyberFlood collects, organizes, manages, and visualizes several global flood databases for the decision makers and public users in real-time. This database is expanded by applying a methodology for the data collection that allows the public to report new flood events using smartphones or web browsers. Hu et al. (2015) implemented a web-based application in the Hadoop-based cloud computing environment to make enhanced coupled human and natural models publically available. This allows users to access and execute the model without an increase in responding time. Kurtz et al. (2017) presented a stochastic cloud-based fully-operational architecture for a real-time prediction and management system related to the groundwater management. This proposed system allows for data assimilation and is coupled with a physically based hydrologic model, HydroGeoSphere, in a cloud environment to use the generated prediction for the groundwater management. This work advances on prior work by demonstrating the ability of using resources in a public cloud, including instances with powerful GPUs like those provided by AWS, to build an end-to-end automated cloud-based system for regional-scale flood forecasting. This system is able to run a computationally-expensive 2D hydrodynamic model and is activated automatically during extreme weather events by software that is continuously monitoring forecasted rainfall conditions for potential extreme events. It is also able to run in a time frame relevant to real-time emergency management applications and automatically delivers model outputs through online maps and emails directly to decision makers.

The remainder of the Chapter is organized as follows. First, a Study Area section is presented to introduce the region where the model is applied. Second, the Data and Methods section is presented to outline the available data sources, the pre-processing steps used to translate this data for use in the model, steps taken to speed-up the model, and the post-processing steps

used to automate the model output dissemination. Next, the Results and Discussion section presents a prototype of the software and the results from applying the system for an extreme weather event. Finally, the Conclusion section provides a summary of the key research outcomes and steps that could be taken to further advance this work.

3.2 Study Area

The study area is in the portion of the Chowan River basin that is within VDOT's Hampton Roads District and is approximately 5,780 km^2 (2,230 mi^2) (Figure 3.1). The study area includes the Meherrin, Nottoway, and the Blackwater Rivers. The longest flowpath along NHD flowline features is approximately 175 km (109 mi) with a slope that varies from nearly 0% to 21%. The study area includes 493 georeferenced VDOT bridges and culverts. Due to a high portion of the study area consisting of low-relief terrain, especially in the eastern part of the study area (Figure 3.2), R^2S^2 utilizes a two-dimensional (2D) hydrodynamic model called Two-dimensional Unsteady Flow (TUFLOW) (Syme, 2001) (https://www.tuflow.com/). The upstream portion of the project domain is modeled using the Hydrologic Engineering Center-Hydrologic Modeling System (HEC-HMS), a lumped hydrology model that is less computationally intensive, in order to generate inflow boundary conditions for the study area. Including these upstream watersheds, the project domain is approximately 11,000 km^2 (4,240 mi^2).

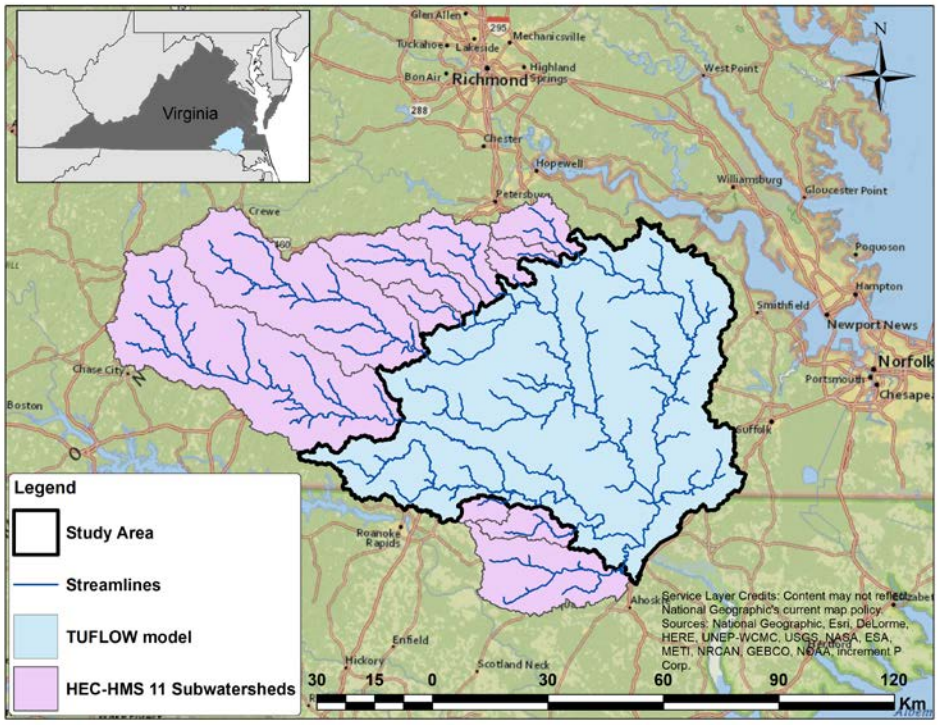


Figure 3.1 Model domain composed of the study area where the TUFLOW model is run and the 11 subwatersheds that contribute inflow to the study area.

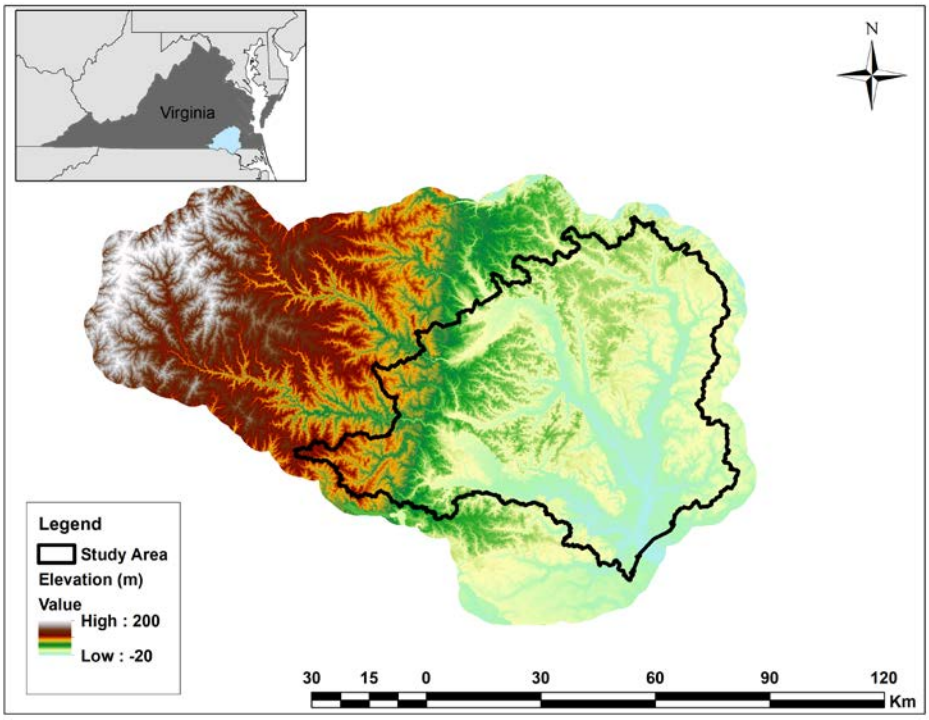


Figure 3.2 The digital elevation model (DEM) with resolution of 10 m x 10 m for the study area including 11 subwatersheds that contribute inflow to the study area.

3.3 Data and Methods

3.3.1 R^2S^2 System

The R^2S^2 system was first developed by Hassan Water Resources, PLC to integrate multiple datasets with sophisticated hydrodynamic models to provide flood risk prediction during severe storm events to the Hampton Roads District of the Virginia Department of Transportation (Hassan Water Resources PLC., 2012). The R^2S^2 system consists of software to process the many input files required for the TUFLOW model, to run HEC-HMS to establish boundary conditions for TUFLOW, and to process output files from TUFLOW to determine inundated bridges and culverts (Figure 3.3). Input data for the R^2S^2 such as DEM, soil, and land use data are constant, however the observational data must be processed in real-time from federal data providers. R^2S^2 uses real-time forecast products for rainfall. The rainfall data is used as inputs for both R^2S^2 hydrologic models. First, the HEC-HMS model uses the rainfall data to generate inflow time series for each of the 11 subwatersheds that border the study area. The TUFLOW model requires these 11 inflow time series as boundary conditions as well as the raw rainfall data to be executed and generate water levels throughout the study area. Historic stream gauge data is used to calibrate and verify the model and, eventually, real-time stream data will be used to set initial conditions.

3.3.2 Rainfall Forecast Data Automation and Preparation

In this study, the procedure to collect and process the rainfall data for model input was automated to reduce human translation errors and decrease the time between when new rainfall forecasts are available and new water level forecasts can be generated. Both the TUFLOW and HEC-HMS models in R^2S^2 require input rainfall data, but in different formats. TUFLOW has three approaches for applying the rainfall directly to the computational cells: (i) polygons covering

multiple cells assigned as rainfall time series, (ii) gridded rainfall created as ASCII files for each time step or as one NetCDF file, and (iii) a rainfall control file that allows a user to specify point time series over the model domain and specify how the rainfall is interpolated to the model cells. HEC-HMS has two approaches for applying the rainfall data: (i) a rainfall time series for each basin stored in a data storage system (DSS) file that is prepared by HEC-DSSVue, a program for viewing, editing, and manipulating DSS files (CEIWR-HEC, 2009), and (ii) gridded rainfall that is prepared by HEC-GridUtil, a utility program for managing gridded data with HEC-DSS (Steissberg and McPherson, 2011).

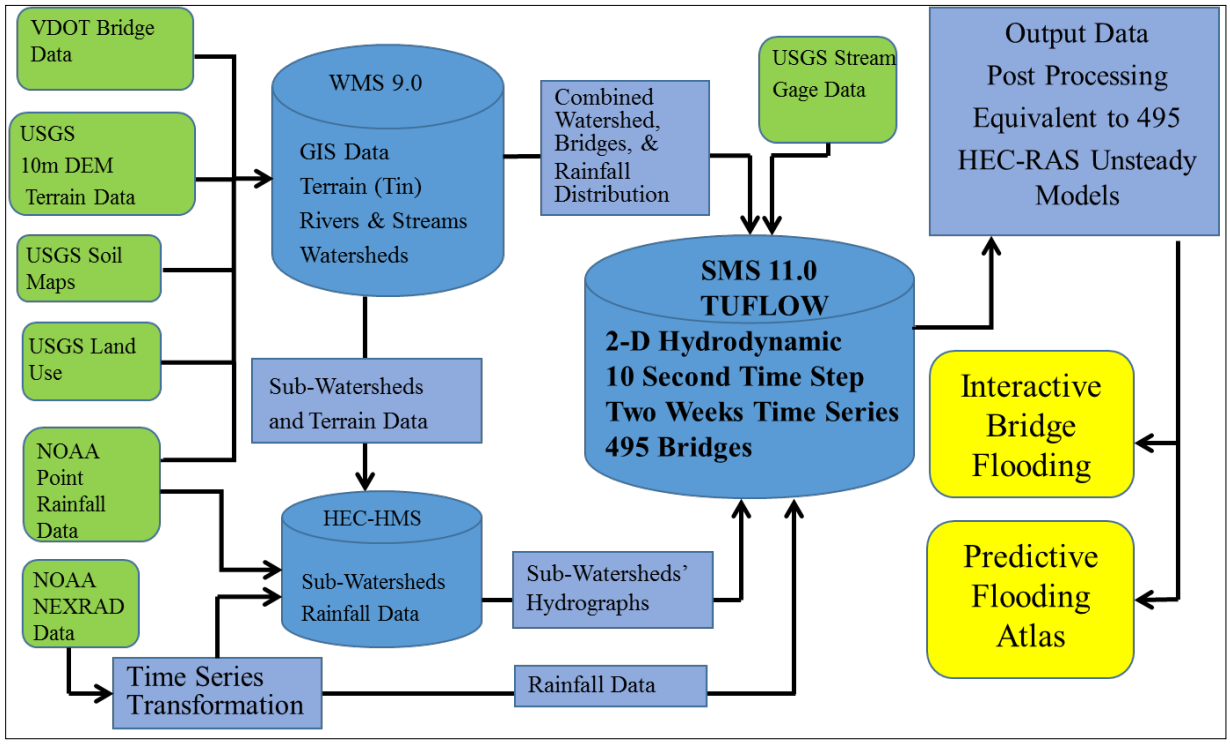


Figure 3.3 R^2S^2 workflow.

The identification of appropriate forecast datasets focused on National Oceanic and Atmospheric Administration (NOAA) products that provides gridded rainfall and can be quickly accessed for real-time flood warning applications. Several potential forecast datasets were identified for the study region. The Rapid Refresh (RAP) product, the High-Resolution Rapid

Refresh (HRRR) product, and the North American Mesoscale Forecast System (NAM) product are all provided by the National Center for Environmental Prediction (NCEP). The National Digital Forecast Database (NDFD) is provided by the National Weather Service (NWS). These forecast products were compared in terms of their spatial resolution, temporal resolution, and frequency of model initiation (i.e. model cycle). Results of this comparison and the code written to automate the workflow for downloading and reformatting the rainfall data to meet the requirements of the TUFLOW and HEC-HMS models are presented in Section 3.4.1.

3.3.3 Speeding-up R^2S^2 Execution

TUFLOW is the computational bottleneck within the overall R^2S^2 workflow. Using a single central processing unit (CPU) for computation takes more than three days to run for a 15 day simulation period (the duration over which Hurricane Sandy caused high flows in the Study region). The use of multiple CPUs and GPUs has been investigated as a means of speeding-up 2D hydrodynamic models (Kalyanapu et al., 2011; Brodtkorb et al., 2012; Rostrup and Sterck, 2010; Castro et al., 2011; Lacasta et al., 2013; Bret et al., 2010; Garcia et al. 2015). As stated in the introduction, using GPUs offers the performance of smaller clusters at a much lower cost (Jacobsen et al. 2010). Therefore, GPUs were investigated for speeding-up the TUFLOW model.

TUFLOW comes with a GPU Module capable of operating on multiple GPUs in parallel. We explored the use of both local and Amazon Web Services (AWS) resources for GPU computations. The TUFLOW GPU Module uses an explicit scheme, while the TUFLOW CPU (TUFLOW Classic) solver uses an implicit scheme. It is well known that explicit schemes are less numerically stable compared to implicit schemes, so the differences between these two schemes could be large and needs to be checked for consistency.

Two local GPU resources with different capabilities were explored (Table 3.1). M1 is a machine with a modest GPU and other resources typical of most desktop computers. M2 is a high-end workstation with 64 GB of RAM and two NVIDIA GeForce Titan Graphics cards. There are several types of AWS Elastic Compute Cloud (EC2) instances designed for GPU-based computations. There are two sizes of G2 instances, which have lower end GPUs, and three sizes of P2 instances, which have higher end GPUs (Table 3.2). The properties and hourly fee for these instances varies as shown in Table 3.2.

Table 3.1 Local computers with GPUs used to investigate TUFLOW model execution times.

ID	Type	CPU	RAM (GB)	GPU	GPU RAM
M1	Desktop Dell OptiPlex 990	3.40 GHz, 4 Core(s)	16	NVIDIA Quadro K2000	2.00 GB, 384 SMX CUDA parallel processing cores
M2	Desktop Viz Lab ESCHER	3.20GHz, 3201 Mhz, 6 Core(s)	64	Two units of NVIDIA GeForce GTX TITAN	6.00 GB, 2688 CUDA parallel processing cores for each

Several tests were performed to measure the TUFLOW model execution times using the AWS EC2 g2.8xlarge and p2.8xlarge instances. The TUFLOW model with 50m grid cell size was used for these tests. The g2.8xlarge instance, which has 4 GPUs, was used to execute the model with 1, 2, 3, and all 4 GPUs. Likewise, the p2.8xlarge instance, which has 8 GPUs, was used to execute the model with 1 through 8 GPUs. Each of these model runs were performed twice to ensure that model run times were consistent.

Table 3.2 Comparison between G2 and P2 EC2 instances performance and cost as of 06/06/2017.

EC2 Instance	Model	GPUs	vCPU	Memory (GiB)	GPU Memory	Storage (GB)	Hourly Fee
G2	g2.2xlarge	1	8	15	4 (GB)	SSD 1 x 60	$0.767
	g2.8xlarge	4	32	60	16 (GB)	SSD 2 x 120	$2.878
P2	p2.xlarge	1	4	61	12 (GiB)	EBS	$1.084
	p2.8xlarge	8	32	488	96 (GiB)	EBS	$8.672
	p2.16xlarge	16	64	732	192 (GiB)	EBS	$17.344

G2 Instances includes NVIDIA GRID K520 GPUs, P2 Instances includes NVIDIA K80 GPUs

3.3.4 Post-processing and Automating Model Output Dissemination

The TUFLOW model computes the maximum water level at each computational cell within the study area throughout the simulation duration. Using these maximum water levels and the VDOT bridge locations and deck elevations, a post-processing workflow was created to automate sending an email with the bridges expected to be overtopped based on model projections. Web resources such as Google Maps and Geosheets (https://www.geosheets.com/) were used to provide real-time visualization for the flooded bridges in the Hampton Roads District. Google Maps has the capability to generate a simple visualization of uploaded keyhole markup language zipped (KMZ) files, which is a quick and simple method to visualize the flooded bridge locations. Geosheets, an add-on to Google Sheets, simplifies visualization capabilities in Google Maps compared to using only the Google Maps application programming interface (API) directly. Using the capabilities of the Google Maps API and the Geosheet application in the post-processing workflow, advanced real-time visualization of the flooded bridge locations can be generated.

3.3.5 Design of an Automated Flood Warning System through AWS

After automating the retrieval of the forecast rainfall data, speeding-up the 2D model, and providing methods for warning decision makers and visualizing the overtopped bridges and culvert

locations in the study area individually, the final step was to create a seamless workflow in AWS to link these three components together without the need of intermediate human action. The goal of this automated workflow is to identify the flooded bridges and/or culverts in a time duration that is sufficient for warning and emergency management purpose based on the highest resolution official rainfall forecast data available. To accomplish this, the design had to meet the following requirements. Given that a single instance capable of all of these tasks would be too expensive to continuously run, a smaller, low cost instance was used to monitor the rainfall data for upcoming extreme events and trigger a larger instance when a flood event is forecasted. This smaller instance also assumes the role of maintaining the website to display and disseminate the model output runs so that it can be available continuously. Storage resources are needed to archive the processed rainfall data used as model input and the model outputs is needed so that model results can be analyzed at a later time. Finally, a larger instance with NVIDIA GPU capabilities was needed to accommodate the hydrologic models and be able to run scripts used to prepare the rainfall data input for the models. This instance is also needed to disseminate the output data. To automate these steps of the workflow, the GPU instance needed to execute a batch file that (i) runs the pre-processing scripts to prepare the rainfall data, (ii) runs the hydrologic models, (iii) runs the post-processing script for preparing the model output for dissemination, (iv) sends outputs to other cloud resources for archiving and visualization, and (v) removes the model output files from the GPU instance.

3.4 Results and Discussion

3.4.1 Rainfall Forecast Data Automation and Preparation

Comparison of the spatial resolution, temporal resolution, and model cycle of each dataset (Table 3.3) shows that HRRR was the best forecast rainfall product for our purpose. HRRR is a weather prediction system composed of a numerical forecast model and an analysis/assimilation system to initialize the model. HRRR is a higher-resolution model nested inside the hourly updated RAP. Although RAP can provide upper-level analyses and short-range forecasts, HRRR is best used to examine surface and near-surface parameters, such as surface precipitation. The HRRR model is run every hour of the day and forecasts out to 18 hours on a one hour time-step for each cycle. It provides a surface total precipitation product in units of mm of precipitation depth at a horizontal resolution of 3 km (NOAA, 2012). Surface total precipitation can be accessed as gridded data with dimensions of longitude, latitude, and time. Longitude and latitude are provided in the World Geodetic System (WGS) 1984 coordinate system, and time is in units of decimal days since 1-1-1 00:00:0.0 (NOAA, 2017a). HRRR data are distributed as a part of the NOAA Operational Model Archive and Distribution System (NOMADS) project, a network of data servers that use the Open Source Project for a Network Data Access Protocol (OPeNDAP) (NOAA, 2017a). Although the HRRR data is used as the primary input to the model, the system could use the coarser Quantitative Precipitation Forecast (QPF) from the NDFD dataset, which forecasts rainfall for the next 72 hours, to monitor for large rainfall events beyond the 18 hours horizon captured by HRRR. This would allow for a longer lead time for preparing for severe storms. This system is developed in a flexible way to enable the use of higher resolution rainfall forecast data that may be available in the future. The use of better rainfall forecast data with a longer lead time will reduce the uncertainty of the model, making it a more useful decision support tool.

Table 3.3 Comparison of available forecast datasets.

Dataset	Data Provider	Relevant Data Product	Resolution		Forecast (hrs)	Model Cycle
			Spatial (km)	Temporal (hrs)		
HRRR	NCEP	Surface total precipitation	3	1	18	24/day
RAP	NCEP	Surface total precipitation	13	1	18	24/day
NDFD	NWS	Quantitative precipitation forecast	5	6	72	8/day
NAM	NCEP	Surface total precipitation	12	1	36	4/day

Figure 3.4 shows the workflow for downloading and reformatting the forecast rainfall data. Pydap, a pure Python library implementing the OPeNDAP, is used to retrieve the desired forecast data for the study area. The automated workflow consists of three main parts: (i) access the latest available forecast data from the HRRR database, (ii) retrieve the forecast surface total precipitation with a horizontal resolution of 3 km x 3 km in WGS 1984 coordinate system, and (iii) reformat the forecast data for model input in the NAD83 UTM 18N projected coordinate system. These rainfall data are reformatted in two ways: gridded rainfall data for TUFLOW using the Geospatial Data Abstract (GDAL/OGR) Python library and subwatershed time series for HEC-HMS using HEC-DSS, Python, and Java libraries. To include these direct rainfall data in TUFLOW, a TUFLOW Event File (TEF) was created to define the storm event properties. For example, using the new TEF file, the user can run the model for a given storm event using either historic or forecast data.

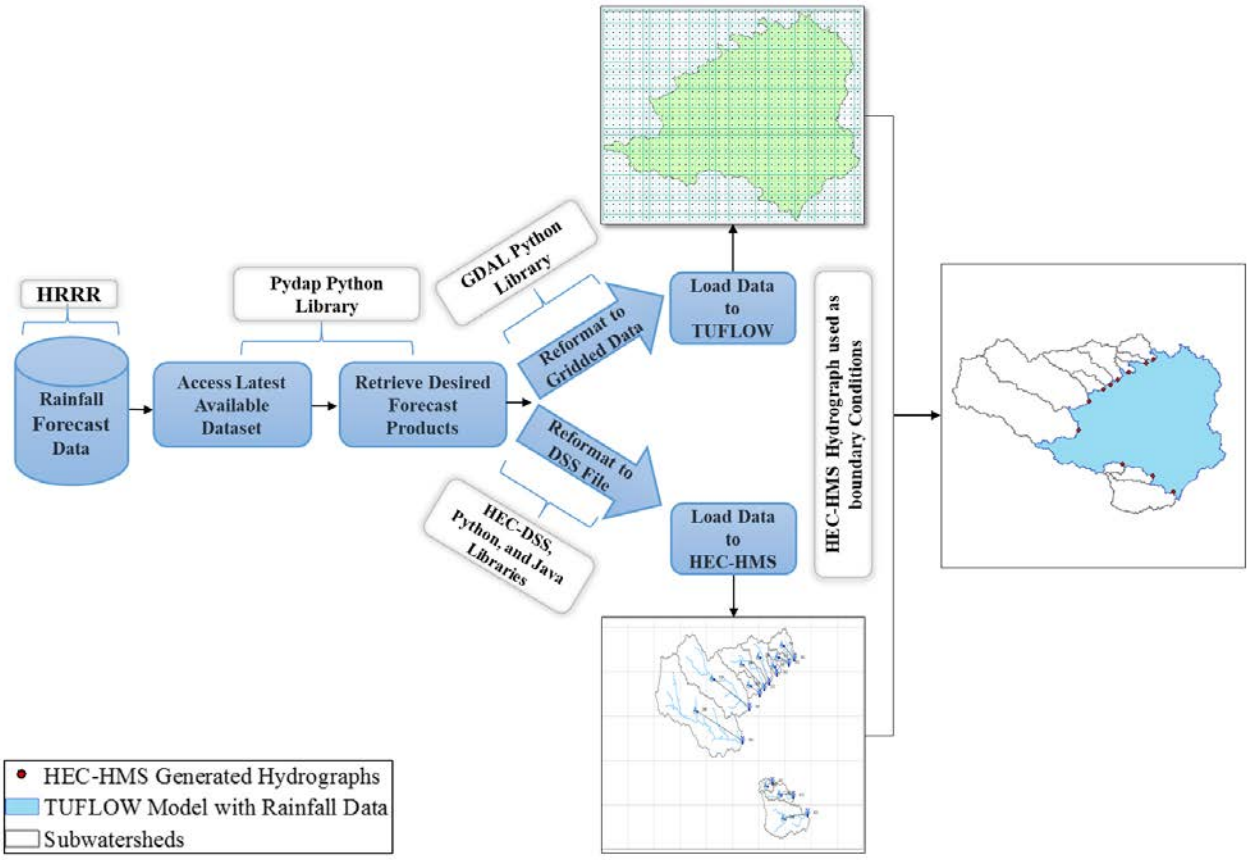


Figure 3.4 Forecast data workflow from HRRR to R^2S^2 sub-models.

3.4.2 Speeding-up R^2S^2 Execution

The speeding-up of the model was performed with model runs using rainfall from Hurricane Sandy as input. The rainfall lasted for four days and the total modeled time span was 15 days (October 28 - November 11, 2012). Table 3.4 summarizes the results of the three TUFLOW model scenarios using the M1 and M2 machines (Table 3.1). Using the CPU, the model took 120 hours to execute. Using the modest GPU in the M1 machine, the model took 11.5 hours to execute (a 10x speed-up compared to the CPU). Using the two more powerful GPUs in the M2 machine, the model took only 2.4 hours to execute (a 50x speed-up compared to the CPU and 5x speed-up

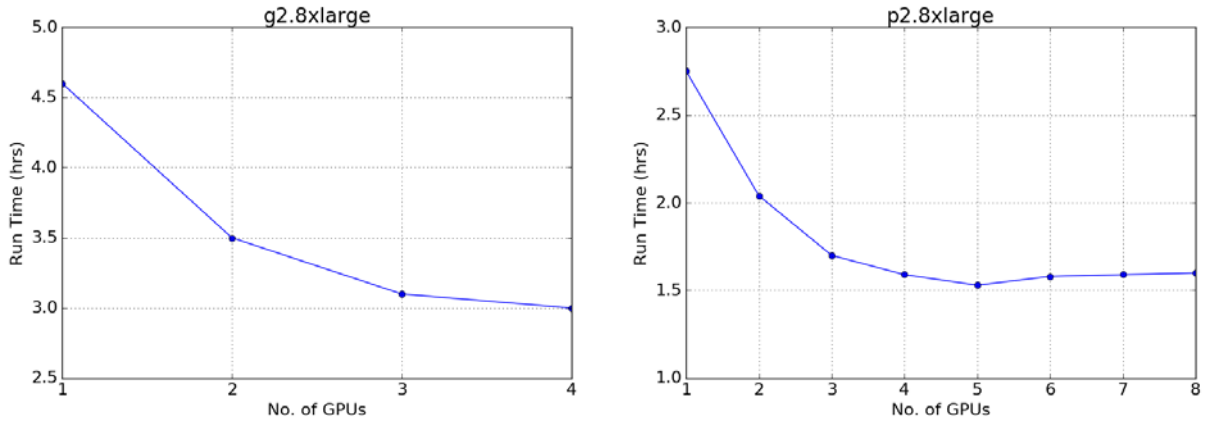
compared to the M1 machine using the single GPU). The input time-step does not have a significant effect on the execute time when using GPUs. This is due to the explicit scheme within the TUFLOW GPU module that takes the user's input time-step value as an initial value and then optimizes the time-step to meet the convergence condition (i.e. courant number <=1) (BMT WBM, 2016) .

Table 3.4 Comparison of CPU versus GPU speed-up using local GPU resources (differences bolded in each scenario).

Model Specifications	**Run Scenarios**		
Machine	M1	M1	**M2**
Processing Unit	CPU	**GPU**	**GPU**
No. of GPUs	-	**1**	**2**
Time-step (sec)	10	10	10
Output Cell Size (m)	25	25	25
Running Time (hrs)	**120**	**11.5**	**2.4**

A test was conducted to determine how increasing the number of GPUs influenced model execution time (Figure 3.5). As expected, running the model by using different numbers of GPUs produced the same output results (i.e., no differences in the maximum water levels). Figure 3.5-a provides the results of this test using the GPU model and the AWS g2.8xlarge instance with different numbers of GPUs. Using the g2.8xlarge instance with one GPU, the model takes about 4.6 hours to run. Using the g2.8xlarge instance and increasing the number of GPUs, the minimum execution time is 3 hours when all four GPUs are used, which costs about $9 per run. Because only four GPUs were available on this instance, we were not able to test whether additional GPUs would continue to reduce the running time. Figure 3.5-b provides the results of this test using the GPU model and the AWS p2.8xlarge instance with different numbers of GPUs. Using the p2.8xlarge instance with one GPU, the GPU model takes 2.75 hours to run, which is less than

using the g2.8xlarge instance with 4 GPUs. This shows the benefit of the more modern GPUs in the P2 versus G2 EC2 instances. Using the p2.8xlarge instance and increasing the number of GPUs, the minimum execution time was found to be 1.5 hours, which is achieved when five GPUs are used. This run, with the minimum execution time (1.5 hours), costs about $13 per run, which is about 1.5x more expensive than the g2.8xlarge instance run; however, it is faster than the g2.8xlarge by 2x. Comparing this 1.5 hours execution time to the CPU execution time of 120 hours shows an 80x speed-up for the model. Using six or more GPUs on this instance increases the execution time compared to using five due to known tradeoffs caused by data transfers between parallel GPU units (Huxley and Syme, 2016).



(a) (b)

Figure 3.5 Running TUFLOW model through AWS EC2 (a) g2.8xlarge instance, and (b) p2.8xlarge instance with different numbers of GPUs.

Because the CPU and GPU TUFLOW solvers use different numerical schemes, it is important to understand differences in their outputs (Figure 3.6). Figure 3.6 provides the differences in maximum water level (Max. WL) generated from executing the model using the CPU and the GPU solvers. The maximum difference in Max. WL across the study area was around 2.5 m (8 ft), with 87% of the computational cells having differences in the maximum water level

less than 0.5 m (1.6 ft). Figure 3.7 shows the Max. WLs at each bridge location generated by executing the model using the CPU solver versus the GPU solver. The mean absolute error (MAE) of 0.48 m (1.6 ft) and the root mean square error (RMSE) of 0.78 m (2.6 ft) demonstrate a fairly significant difference. In this study, we did a preliminary sensitivity analysis by changing the model grid cell size and Manning coefficient values, but future research should investigate this difference more fully.

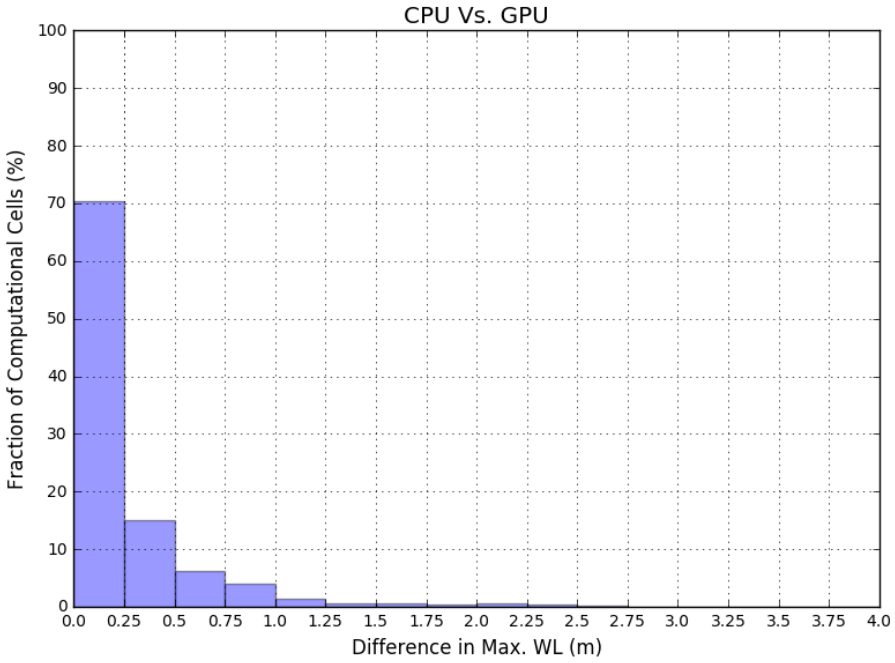


Figure 3.6 Differences between Max. WL generated from CPU solver and GPU solver.

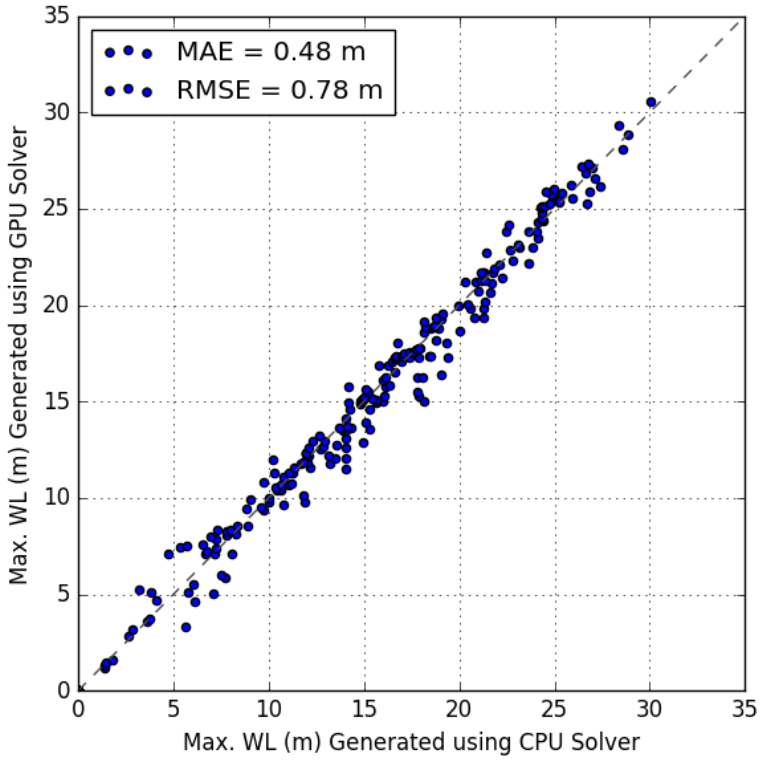


Figure 3.7 Bridges and culverts location Max. WL generated from CPU solver versus GPU solver with MAE of 0.48 m and RMSE of 0.78 m.

The model results using both the CPU and GPU solvers were compared against stream stage observations for Hurricane Sandy event. Figure 3.8 and Table 3.5 show the USGS stations with data availability for the event. The USGS provided unpublished stage data that is considered provisional and, therefore, may contain erroneous or missing values due to instrument malfunction. This data was processed and cleaned to address this issue before being compared to the model output data. Figure 3.8 also shows the NOAA stations with the available recorded rainfall data for the Hurricane Sandy storm event. Hyetographs for this storm event at these stations are shown in Figure 3.9.

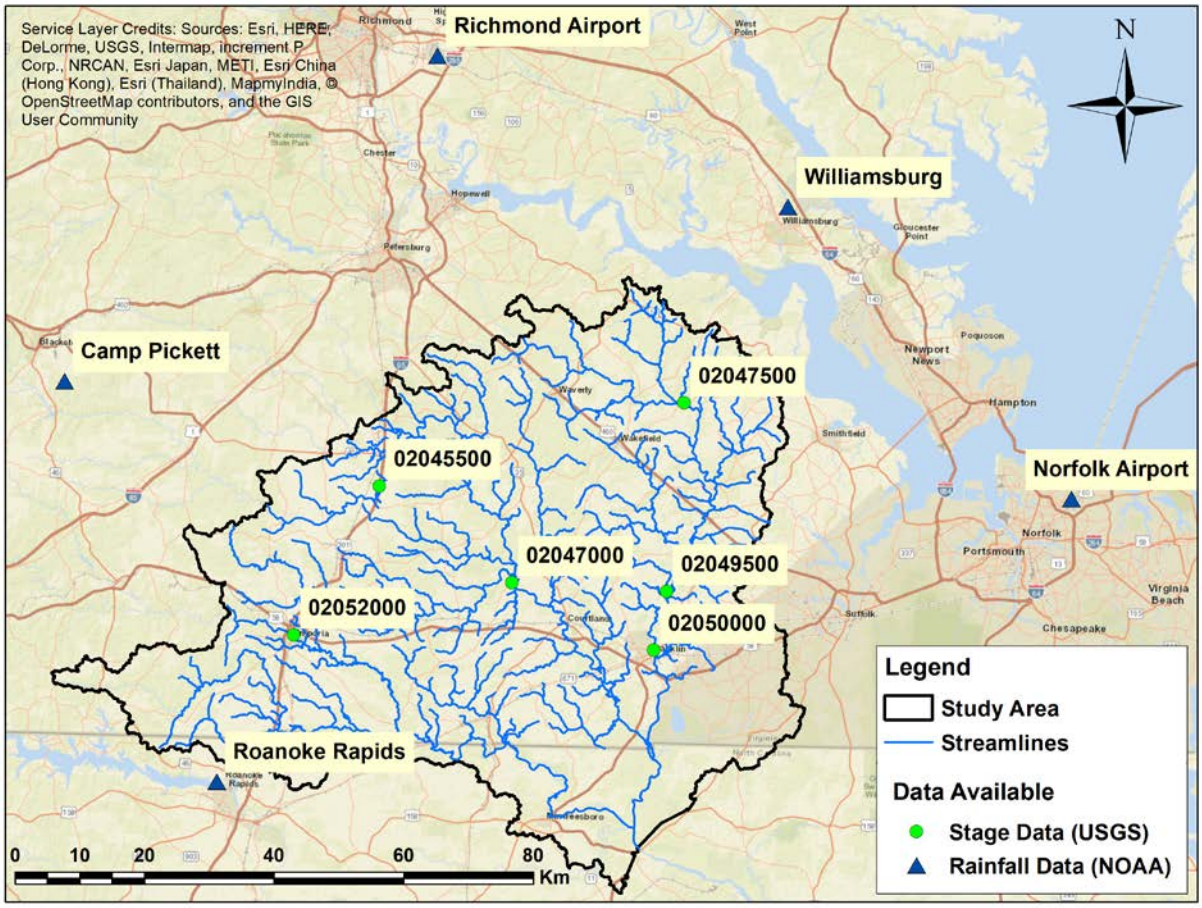


Figure 3.8 USGS and NOAA station locations and Hurricane Sandy data availability in the study area.

Table 3.5 USGS stations in the study area with information about Hurricane Sandy availability.

Station	Name	Current Status	Stage		
			Parameter Code	Start Date	End Date
02049500	USGS 02049500 BLACKWATER RIVER NEAR FRANKLIN, VA	**Active**	00065 Gauge height	10/23/2016	2/20/2017
02047500	USGS 02047500 BLACKWATER RIVER NEAR DENDRON, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02045500	USGS 02045500 NOTTOWAY RIVER NEAR STONY CREEK, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02052000	USGS 02052000 MEHERRIN RIVER AT EMPORIA, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02047000	USGS 02047000 NOTTOWAY RIVER NEAR SEBRELL, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017
02050000	BLACKWATER RIVER AT HWYS 58/258 AT FRANKLIN, VA	**Active**	00065 Gauge height	10/31/2016	2/28/2017

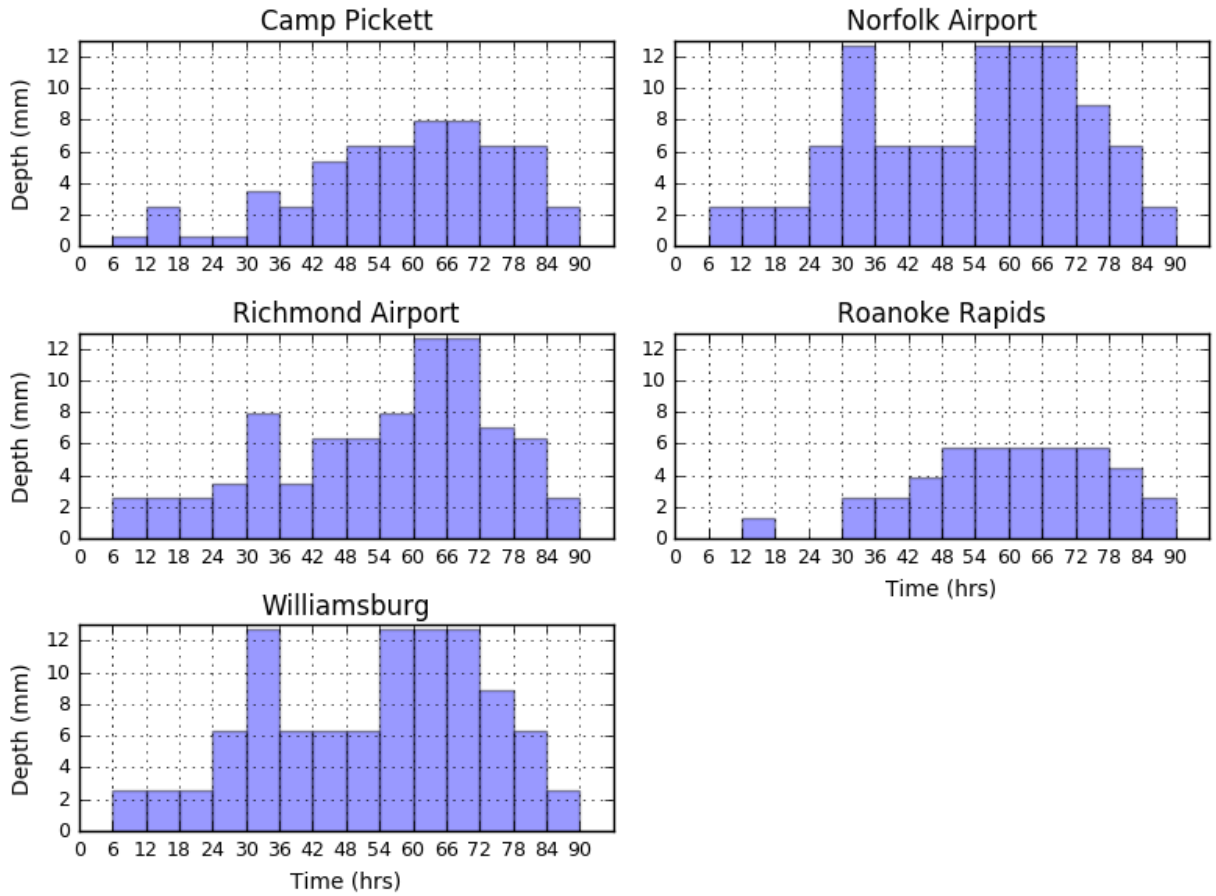


Figure 3.9 Hurricane Sandy hyetographs at the five NOAA stations near the study area (Figure 3.8).

The finite volume schemes used by the 2D models are heavily dependent on the gird cell shape and size (LeVeque, 2002; Caviedes-Voullième et al., 2012). The TUFLOW model GPU solver uses only a Cartesian grid with the capability of changing the grid cell size. The TUFLOW model was executed using the GPU solver with grid cell sizes of 50 m, 40 m, 30 m, and 20 m. The output data from each of these runs was compared to the observed data at the six USGS stations and model results from executing it using the CPU solver with cell size of 50 m. The modeled peaks using the GPU solver with 50 m grid cell size were significantly higher than the observed data and the model peaks using CPU solver at four USGS stations (02045500, 02047000, 02047500, and 02052000). However, at one of the USGS stations (02050000), the modeled peak using the GPU solver with 50 m grid cell size was significantly lower than the observed data and the modeled peak using the CPU solver. Finally, at another USGS station (02049500), the modeled peak using the GPU solver with 50 m grid cell size was almost the same as the model peak using the CPU solver. However, both peaks were significantly lower than the observed data.

The differences between the modeled and observed peak stages could be due to the lack of adequate bathymetry data in the major rivers and tributaries. In all of the minor tributaries and some stretches of the main rivers, bathymetry had to be assumed because no bathymetry data was available. This also could be due to the coarse DEM resolution (10 m x 10 m). Calibration with limited data available for such a large study area is a challenge as well, especially with a scarcity of operating river gauges and available data for event-based calibration. In some instances, 2D models are not used due to the low resolution of the available spatial data and the difficulties faced when calibrating the model parameters (Caviedes-Voullième et al., 2012). This large study area includes only six USGS gauges that recorded stream stage during Hurricane Sandy. Three of these stations are located on the same main stream at the eastern part of the study area, one is in the

middle of the study area, and the other two are located in the western part of the study area. More gauges would be valuable to be more confident in the calibration of such a large 2D model.

When the cell size of the model using the GPU solver decreases, a significant reduction in the peak stages was observed at four of the six USGS stations (02045500, 02047000, 02047500, and 02052000). At station 02050000, the modeled peak stage using the GPU solver increased with decreasing cell size, while at station 02049500 the peak stage remained nearly constant with each cell size. Decreasing model grid cell size improved the matching of observed peaks at four of the six observation sites and, therefore, we decided to use a smaller cell size in the model application. The drawback of a smaller cell size is an increase in model execution time. Figure 3.10 shows the model execution time using the GPU solver with different grid cell sizes (50 m, 40 m, 30 m, and 20 m) for the M2 machine (Table 3.1). Figure 3.10 also shows the MAE generated from comparing the model output using different cell sizes and using the GPUs with the model output using the CPU solver with the 50 m grid cell size. Based on these results, we chose the 30 m cell size since there is only a small difference in the results using the GPU solver with a 20 m grid cell size model and there is a significant increase in the model run time (2.8x from 10.2 hours to 28 hours).

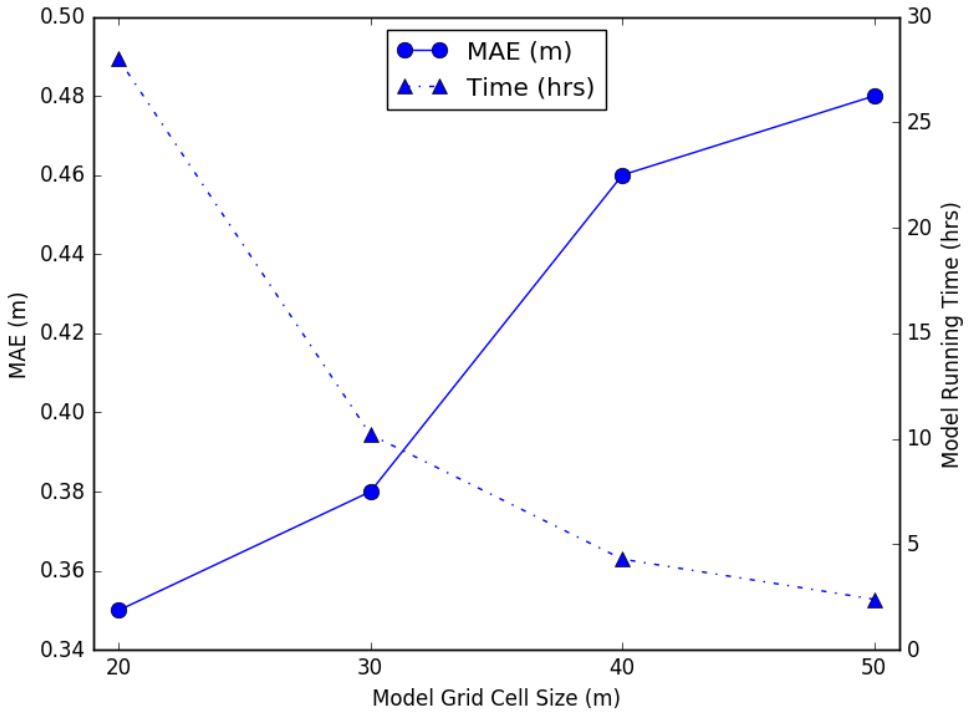


Figure 3.10 Model run time using GPU solver with different grid cell sizes and the corresponding MAE versus CPU solver using M2 machine (Table 3.1).

In addition to decreasing the cell size to 30 m, we also adjusted the Manning coefficient (n) to test its sensitivity and ability to improve matching of observed peak stages obtained from the six USGS stations. The model initially had Manning coefficient values determined based on the study area land use. To assess the sensitivity of the model to changes in the Manning coefficient, this coefficient was changed to be 0.6n, 0.8n, 1.0n, 1.4n, and 1.8n. As the Manning coefficient value decreased, the modeled peak stages became closer to the observed peaks at stations 02045500, 02047000, 20047500, and 02052000. After reducing the grid cell size from 50 m to 30 m and the Manning's coefficient from 1.4n to 0.6n, the model came the closest to matching observed peak river stage. This represents a preliminary calibration of the model that should be more fully explored through additional research.

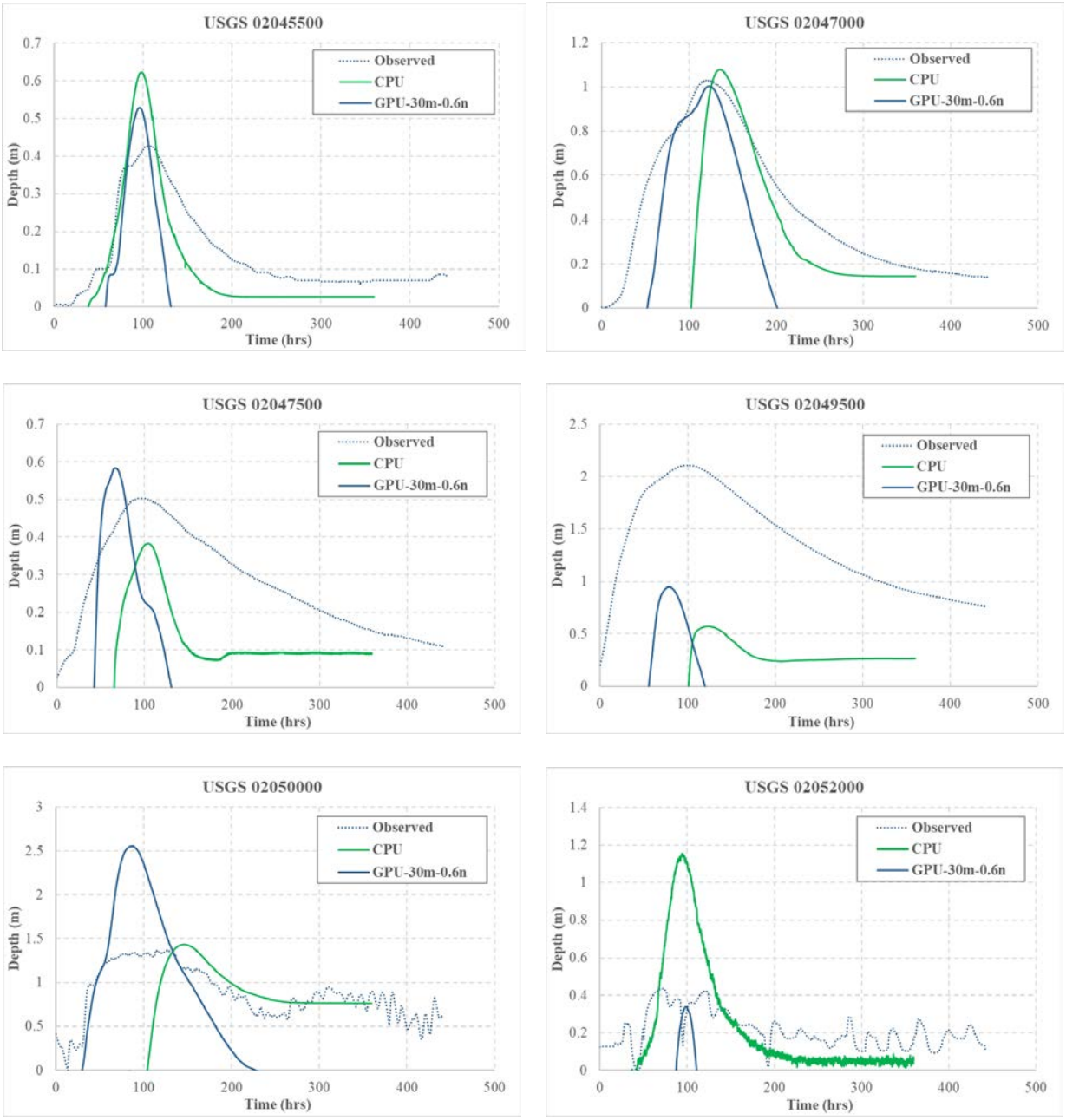


Figure 3.11 Comparison between the stage depth observation data and the output depth from executing the model using a GPU solver with 30 m cell size and 0.6n Manning coefficient values.

The analysis of changing the model grid cell size and Manning's coefficient was done by applying rainfall time series for Hurricane Sandy from five rain gauges to polygons that each covered multiple model grid cells. TUFLFOW also has the capability of using direct rainfall data that applies input rainfall values to every cell in the 2D hydrodynamic model. When the rainfall is

directly applied to the cells, the model routes flow based on the cell topography on a cell by cell basis (Huxley and Syme, 2016). Huxley and Syme (2016) investigated using this new method by applying the direct gridded rainfall data and found that GPU direct rainfall hydraulic modeling can be used as an alternative to runoff-routing hydrology modeling. To check the model behavior using the direct gridded rainfall data method with the chosen grid cell size and Manning's coefficient values, rainfall data from Hurricane Sandy was obtained from the Tropical Rainfall Measuring Mission (TRMM). This data has resolution of 0.25 x 0.25 degrees resulting in 16 cells covering the entire study area. We hoped to use rainfall data from the Next Generation Weather Radar (NEXRAD) provided by NOAA, but there was no data available for the dates of Hurricane Sandy for our study area.

Figure 3.11 shows the results of using the gridded rainfall data provided by TRMM when executing the model with grid cell size of 30 m and Manning's coefficient values of 0.6n using the GPU solver. Using the gridded rainfall data with this coarse resolution produces results very similar to those found when using the rainfall gauge data and the polygon method. The model results almost matches the observation peaks at the 02045500, 02047000, 02047500, and 02052000 USGS stations. The other two USGS stations, 02049500 and 02050000 where the modeled peaks are further from the observed peaks, are located on the same stream at the eastern part of the study area along with Station 02047500. This area has the mildest slopes in the study area (almost flat) (Figure 3.2). The station furthest upstream is 02047500. At this station, the model predicts a slightly higher peak than the observed data and the modeled peak using the CPU model. The second station (02049500) has a much lower peak than the observed data. However, the modeled peak using the CPU solver is even lower than the modeled peak using the GPU solver. The peak at station 02050000 is much higher than the observed peak and the modeled peak using

the CPU solver. The variation between the observed and modeled peaks at these three stations could be due to the coarse DEM resolution (10 m x10 m) used in the model. The slightly higher peak at 02047500 may be due to slopes derived from the DEM being milder than the real slopes. The much lower peak and lower volume at 02049500 could be due to having slopes derived from the DEM much steeper than the real slopes. Like with 02047500, the much higher peaks at 02050000 may be due to the DEM-derived slopes, which are milder than the real slopes. This would explain why the absolute differences in the peaks at stations 02049500 and 02050000 are nearly the same but the one is below and the other is above the observed peak. If the slopes of the contributing areas to station 02049500 were milder, the peak there would be higher and the peak at the downstream station (02050000) would be lower, making both closer to the observed data. This might improve if a higher DEM resolution is used within the model. Future work will explore this and the use of NEXRAD to better understand the benefit of this rainfall data for predicting the stage depth peaks.

3.4.3 Post-processing and Automating Model Output Dissemination

Figure 3.12 shows the resulting workflow for model output post-processing for dissemination of model results. This workflow uses different Python libraries such as GDAL/OGR, Simple KML library (SIMPLEKML), and an email library to generate the visualization of the flooded bridge locations and automatically email warnings to decision makers. The workflow and its products could be used with ArcMap, Google Maps, Google Earth, Geosheets or a custom website, such as the one we configured and hosted on the AWS, EC2 t2.micro instance, as shown in Figure 3.12. There are three products for visualization that can be generated from this workflow: (i) an ESRI shapefile that includes just the flooded bridges, (ii) a

KMZ file that includes flood information for all bridges that can be visualized through Google Maps or Google Earth, and (iii) a dynamic and real-time visualization on Geosheets created by automatically uploading the bridges with their flooded status to a Google Sheet using the Google Drive API. Unlike hosting a website to visualize the KMZ file on the EC2 t2.micro instance, using GeoSheets requires no webserver. However, hosting our own website in the long run will provide much more flexibility and the potential for more capabilities. Figure 3.13 shows an example of an advanced visualization for the flooded bridges directly on the Geosheets permanent URL once the workflow runs. This visualization shows the bridges as being not overtopped (green), nearly overtopped (yellow), and overtopped (red) from forecast rain events.

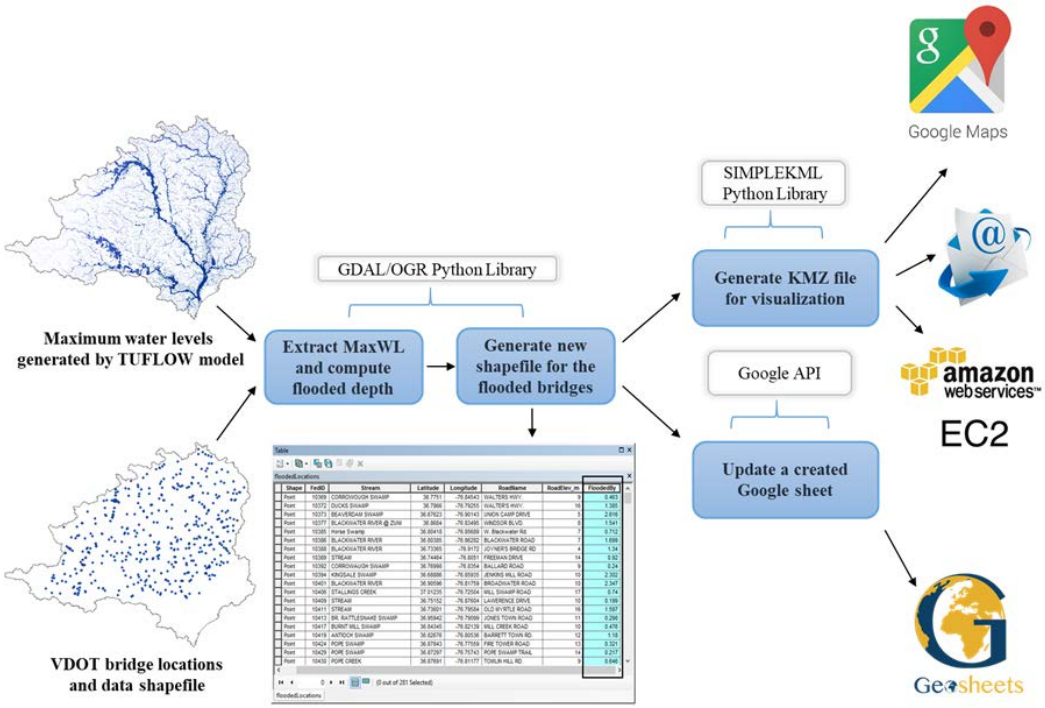


Figure 3.12 Post-processing workflow for producing different visualization resources.

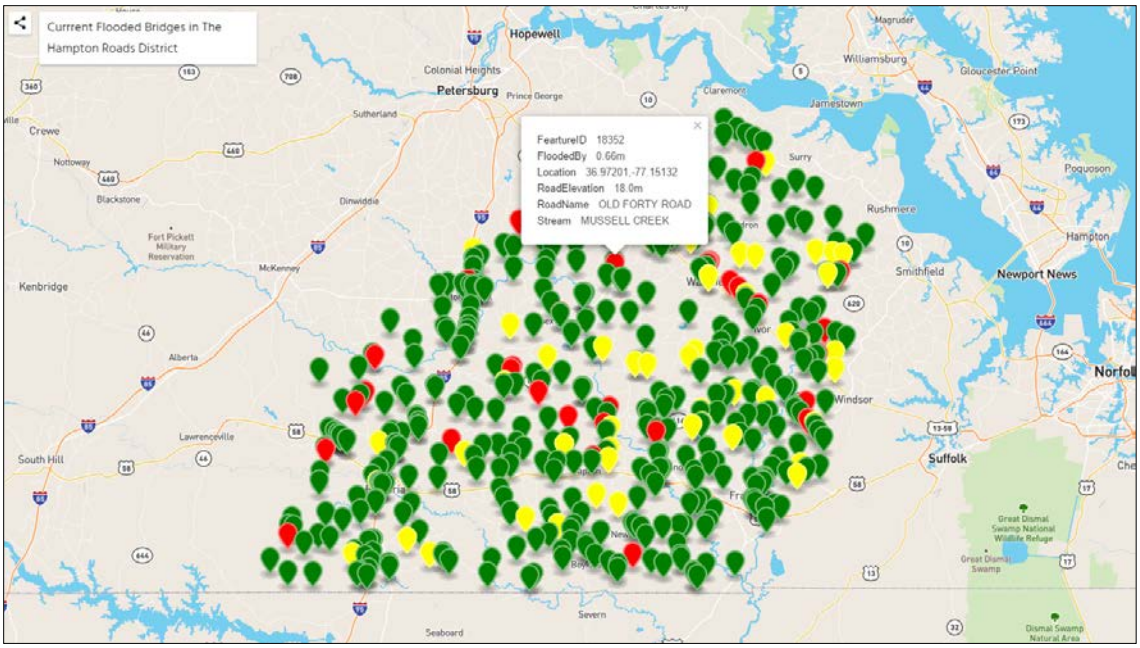


Figure 3.13 Real-time visualization with permanent URL for visualizing the flooded bridges location using Geosheets. (https://www.geosheets.com/map/s:Lo6Wq0Jl/Currrent-Flooded-Bridges-in-The-Hampton-Roads-District).

3.4.4 Automated Flood Warning System through AWS

Figure 3.14 shows the design of the automated workflow that meets the design requirements outlined in the methods section. This solution uses three AWS resources: (i) a low cost EC2 t2.micro instance running a Linux operating system, (ii) an EC2 G2 or P2 instance with Windows operating system, and (iii) a S3 Bucket. The EC2 t2.micro instance has two roles in the workflow. First, the instance continuously monitors rainfall forecasts to identify an extreme weather event. When an extreme weather event is identified, the EC2 t2.micro instance starts the EC2 G2 or P2 instance and a model run is initiated. Second, the EC2 t2.micro instance serves the webpages used to visualize and disseminate the model results computed by the larger EC2 G2 or P2 instance. The EC2 G2 or P2 instance includes all of the model components. The EC2 G2 or P2 instance retrieves, preprocesses and prepares the forecast rainfall data for the hydrologic models.

This same instance also executes the 2D hydrologic model. After the model runs, the EC2 G2 or P2 instance sends model outputs to the EC2 t2.micro instance for visualization and dissemination. The model outputs are also sent, along with the processed forecast rainfall data used as model inputs, to the S3 bucket for archiving and reproducibility purposes.

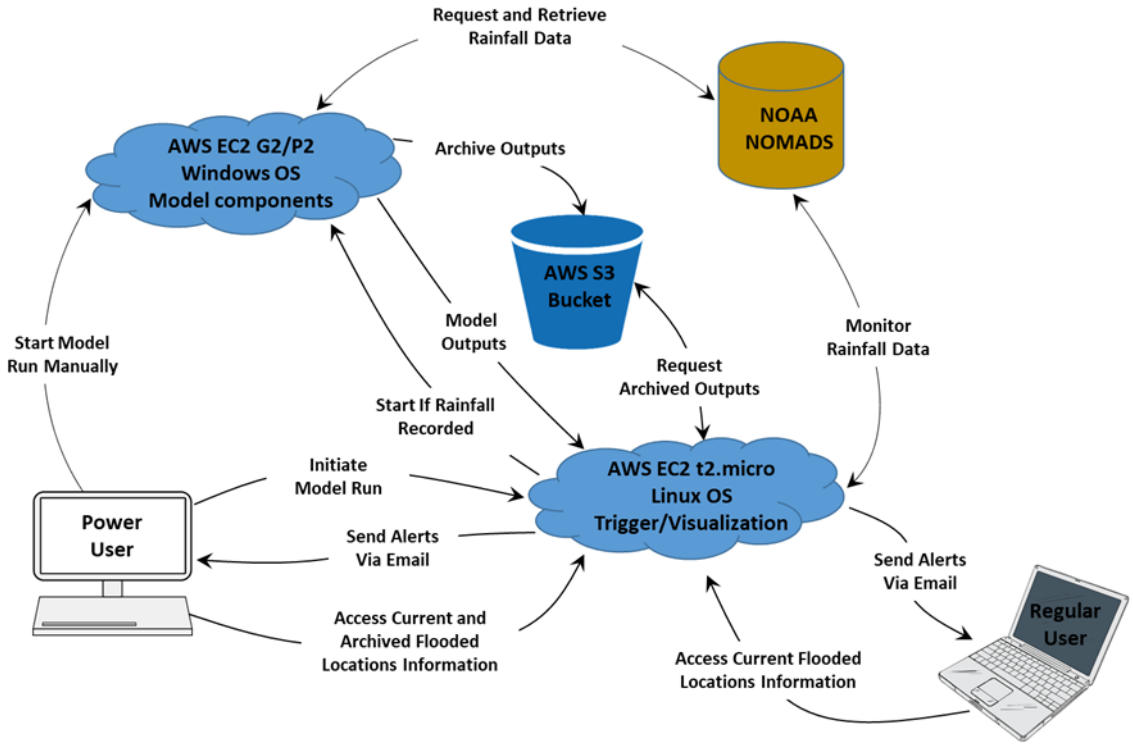


Figure 3.14 Design of the automated workflow for a flood warning system using AWS resources.

There are two classes of users that can access the model outputs via the webpages running on the EC2 t2.micro instance: regular users and power users. Regular users can access the current flooded locations and can register to receive alerts via email whenever locations are forecasted to flood. In the current implementation, regular users do not need to authenticate with the system. Power users have more privileges than the regular users, including access to all the archived

inundation maps from the S3 bucket and the ability to run the model at any time via a powershell script or through the website hosted by the t2.micro instance.

AWS has the ability to securely control access to services and resources for specific users using the Identity and Access Management (IAM) service. This service was used to give permission to the EC2 t2.micro instance to start and stop the other EC2 G2 or P2 instance. A new user was created and then given permission for starting and stopping the EC2 G2 or P2 instance (Figure 3.15). By using the new user credentials, the EC2 G2 or P2 instance ID, and command lines executed in a scripting language or at the AWS command line interface (CLI), the EC2 G2 or P2 instance can be started and/or stopped automatically. The main script in the development web framework on the EC2 t2.micro instance is called server.py. Code was added to this Python script for monitoring and accessing the other EC2 G2 or P2 instance. In this code, a process is run every hour to check the HRRR rainfall data (which is updated hourly). If the forecasted rainfall is over a certain threshold value, it will start the EC2 G2 or P2 instance that includes the hydrologic model automatically. Then the EC2 t2.micro instance keeps monitoring the EC2 G2 or P2 instance to make sure that it is fully started (this is done by adding additional permissions to the user policy). Then the EC2 t2.micro instance uses Secure Shell (SSH) to initiate a batch file that runs the main workflow for retrieving the data, executing the model, and generating the output. The 2D hydrologic model takes about 10 minutes to run through a forecasted period (18 hours) using a model grid resolution of 50 m on the M2 machine, while it takes about 38 minutes using the model grid resolution of 30 m on the M2 machine. The running time for the model with grid resolution 30 m is expected to be lower when using the EC2 p2.8xlarge instance. Using the p2.8xlarge AWS instance with five GPUs, it is expected that the runtime will be 6.3 minutes for a 50 m grid cell size and 24 minutes for 30 m grid cell size.

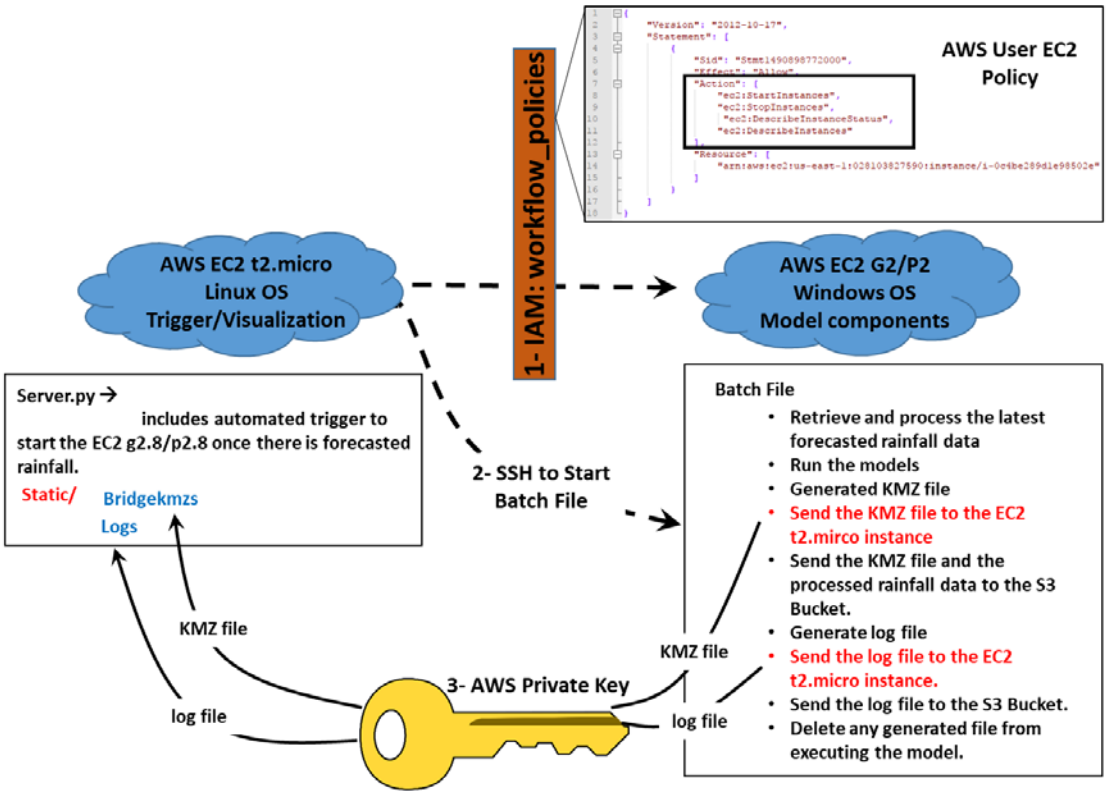


Figure 3.15 The policies between the EC2 t2.mico and G2 or P2 instances.

This batch file that automates the model execution operates as follows. First, the HRRR data is retrieved and processed. Once the rainfall data is retrieved and is available for the hydrologic models, the models are run and the maximum water level at each computational cell within the study area is computed and recorded for the duration of the simulation period. Once the maximum water level output file is available, the KMZ file is generated from the model output file. This KMZ file includes information about each bridge and culvert provided by VDOT, the maximum water level predicted by the model, and by how much each bridge would be overtopped. The KMZ file is sent to the t2.micro instance to be used for visualization. This is done using the AWS Private Key generated for the EC2 t2.micro instance. Another policy added to the IAM user is used to access the S3 Bucket and archive the processed rainfall data (Figure 3.16). A log file is generated that includes a record of the parameters and scripts used in the whole process as a

reference for users or decision makers. The log file is sent to both the EC2 t2.micro instance and the S3 Bucket for archiving. Finally, any files generated from running the whole workflow are deleted to minimize the storage on the EC2 G2 or P2 instance.

A power user can use a powershell script to automatically initialize a model run. The script gives the user the option of running the workflow either locally or with the EC2 G2 or P2 instance. When the workflow is chosen to run locally, the powershell script installs any required dependencies and then runs the batch file to start the workflow. If the user chooses to run the workflow through the cloud, the script asks for the IAM policy credentials and starts the EC2 G2 or P2 instance. Once the instance is fully started, the script uses SSH to run the batch file to start the main workflow.

Figure 3.16 shows the different policies used by the EC2 t2.micro and G2 or P2 instances to access the S3 bucket folder that includes the archive information for each run. Also this figure shows the hierarchy of the S3 Bucket folders for archiving the workflow output data. The S3 Bucket folders receive data from the EC2 G2 or P2 instance once it starts. To give full access for these specific folders and their contents to the EC2 G2 or P2 instance, another policy was added to the IAM user (Figure 3.16). The EC2 G2 or P2 instance uses the IAM user policy to access the main folder, floodwarningmodeldata, and archive the output data generated by the workflow in each specific subfolder. The EC2 t2.micro instance then retrieves the archived KMZ and log files to visualize them on the website. This is done by using a separate policy provided by the AWS S3 Bucket (Figure 3.16).

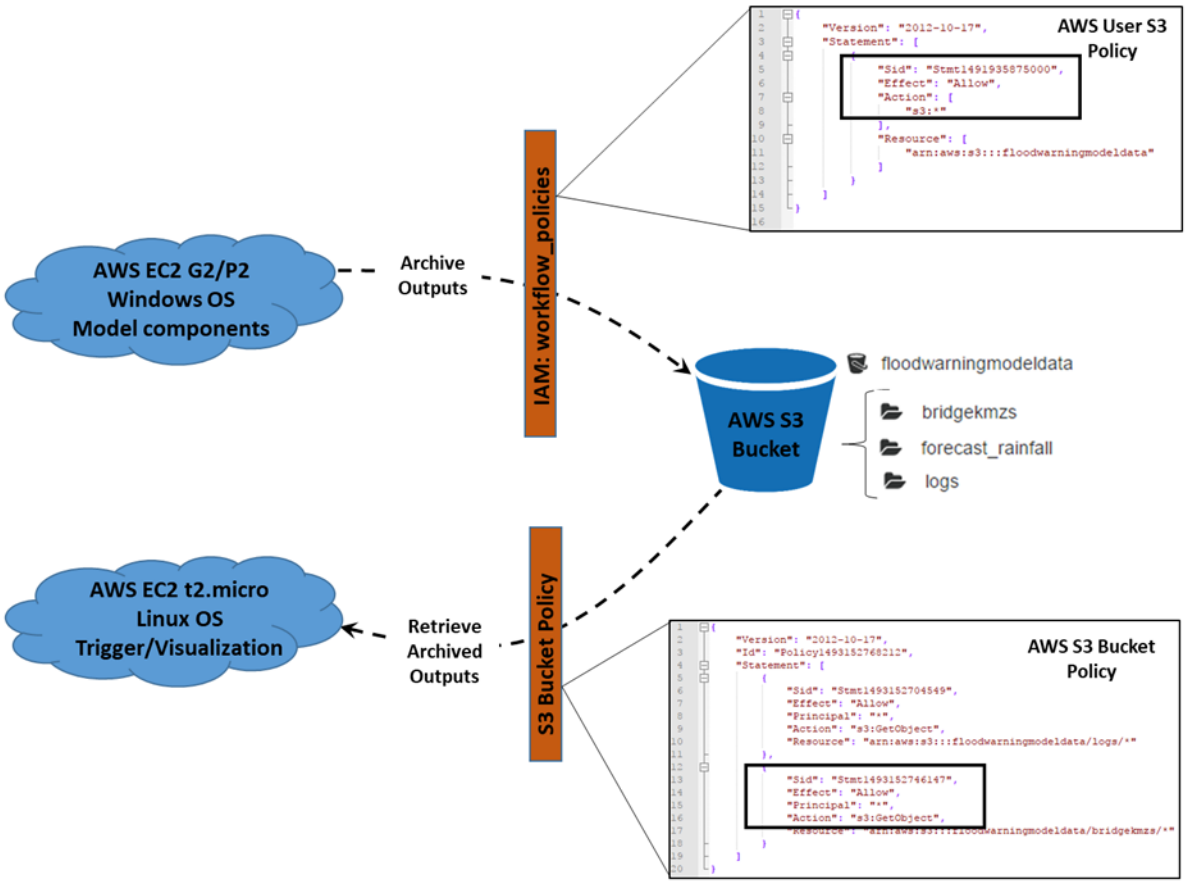


Figure 3.16 Different policies used to access the AWS S3 Bucket data, and the AWS S3 Bucket folder hierarchy.

The t2.micro instance handles the visualization of the output data using a Python based micro web framework, Flask (http://flask.pocoo.org/) (Figure 3.17). When a user accesses the website URL (http://ec2-34-207-240-31.compute-1.amazonaws.com/) the most recent model output KMZ is displayed using the Google Maps JavaScript API. The output KMZ files, along with the corresponding log files from only the five most recent model runs, are available on the website to save storage space. NGINX (https://nginx.org/en/) and Gunicorn "Green Unicorn" (http://gunicorn.org/) sit in between the flask application and the internet working in tandem to support many users on the website at the same time and handle the distribution of resources.

The t2.micro instance also triggers a model run when HRRR rainfall forecast data exceeds a given threshold. The forecast rainfall data is therefore retrieved every hour. If the rainfall exceeds a certain threshold value it will start the EC2 G2 or P2 instance and initialize a model run with the latest rainfall data. An alert on the website will show users whether a model is being run, flooding is possible, or the model is up to date with no flooding predicted.

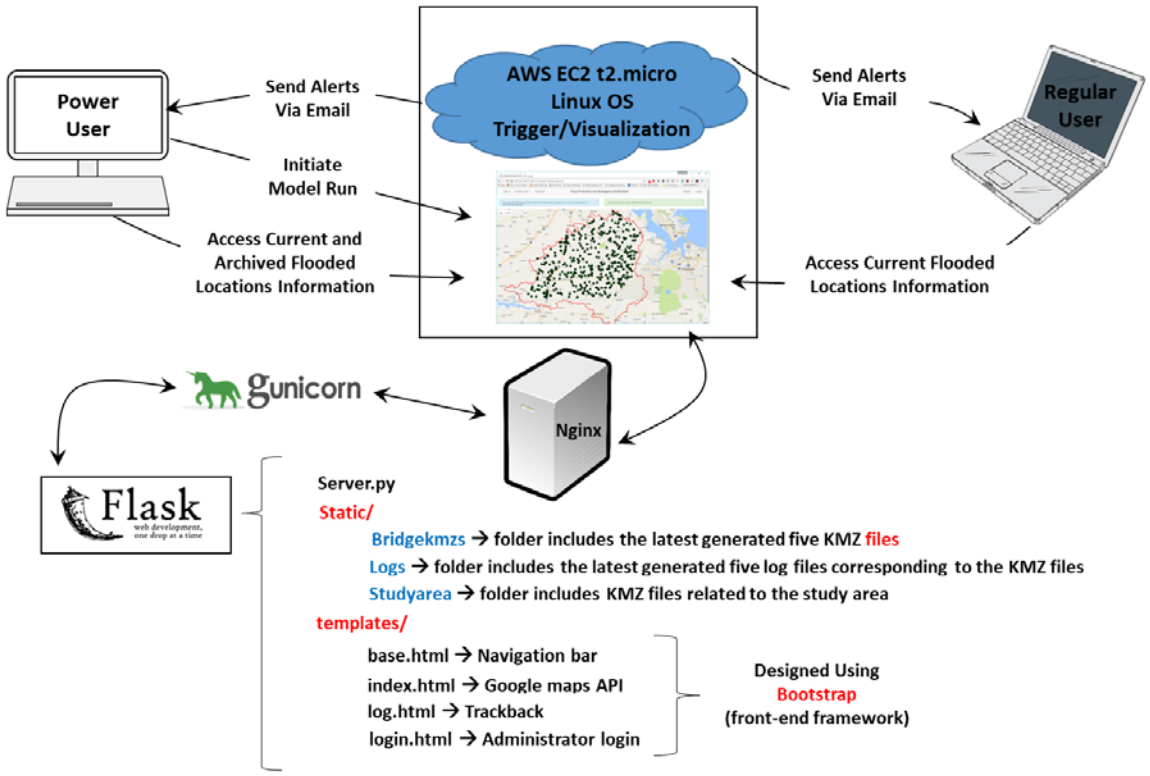


Figure 3.17 EC2 t2.micro instance and the web framework used to build up the website.

Figure 3.18 shows the architecture of the website. On the main view, the website contains a navbar allowing the selection of which data to view, a link to the log file, a login page, or a page to register for email alerts. The main section of the page is taken up by the Google Maps JavaScript API. Using the Google Maps JavaScript API allows us to easily display the map interface using all of Google's resources and overlay our output data on top of it. When a user clicks on a marker signifying a bridge, they are presented with a box containing more information about that bridge

and potential flooding events. Users can sign up and their email will be stored in a secured private Structured Query Language (SQL) database. The application will detect when flooding is possible and send an email to everyone on the list. Through the website, power users can display output data archived in the AWS S3 bucket without having to store output in the t2.micro instance, which has a limited amount of storage.

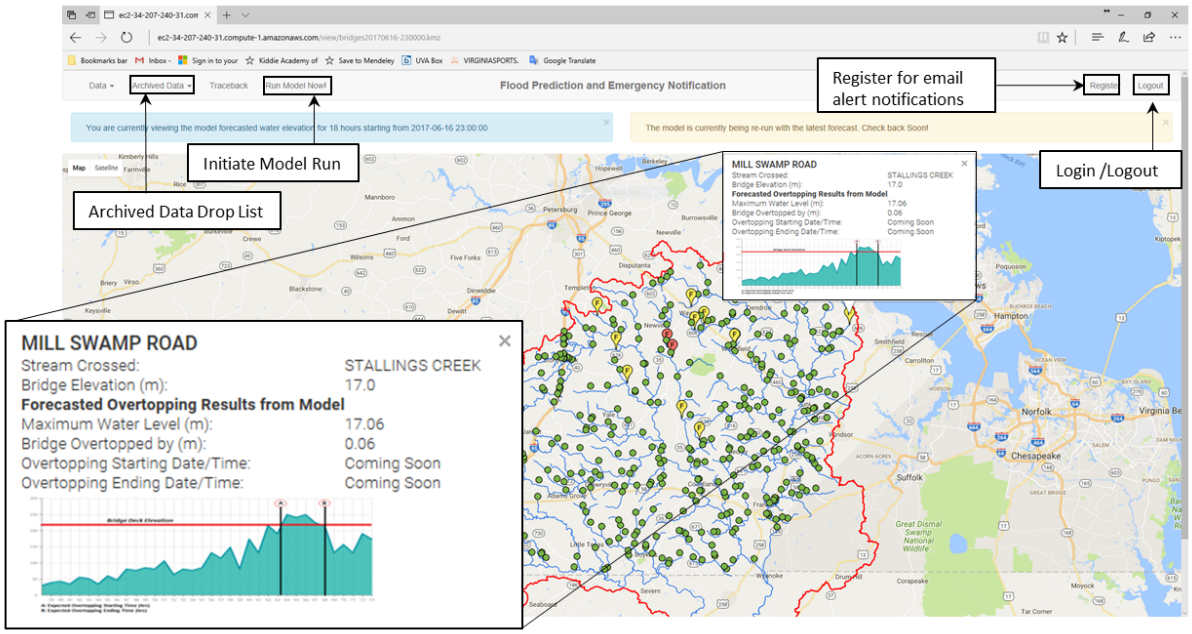


Figure 3.18 Main webpage of the flood warning decision support website.

3.5 Conclusions

This work describes the creation of a cloud-based flood forecasting system designed to assist transportation decision makers in time-sensitive, emergency situations. The flood forecasting system is applied for the Virginia Department of Transportation in the Hampton Roads region to provide decision makers with forecasts of flooded roadways and bridges in near real-time based on rainfall forecasts. By using GPU resources, the model was executed for a 15 day

duration up to 80x faster (from 120 hours compared to 1.5 hours) compared to using a single CPU. An automated cloud-based workflow using AWS resources was designed and created to link and enhance the three core model components: (i) retrieval and formatting of high resolution gridded HRRR rainfall forecast data, (ii) execution of the 2D model in a short duration to identify flood prone bridges and culverts, and (iii) real-time dissemination of model output via generation of an online map with flooded locations and the ability to automatically send alert messages via email.

Using the M2 machine described earlier, the 2D hydrodynamic model, which is the heart of the flood forecasting system, completes an analysis for the upcoming 18 forecast hours in approximately 10 minutes with a model grid cell size of 50 m, and approximately 38 minutes with a model grid cell size of 30 m. Using the p2.8xlarge AWS instance with five GPUs, it is expected that the runtime will be 6.3 minutes for a 50 m grid cell size and 24 minutes for 30 m grid cell size. For Hurricane Sandy, although the rainfall only lasted 4 days, the effects of the rainfall over the study area lasted 15 days. Assuming running the model with a 50m grid cell size takes 6.3 minutes to run for the upcoming 18 forecasted hours on the p2.8xlarge, if the model ran every hour through 15 day period, running the workflow would cost about $350 assuming current AWS prices. For the same scenario but with a grid cell size of 30 m, modeling 18 hours is expected to take about 24 minutes to run and cost $1260 for the 15 day duration. However, this assumes using five GPUs and further tests are required to run the model with grid cell size of 30 m on the AWS EC2 P2.8 instance to find the optimum number of GPUs for this scenario.

Because the TUFLOW 2D model is expensive to run, it is only used during extreme weather events. The t2.micro instance, which costs about $10 per month to run continuously, monitors the HRRR forecast rainfall data comparing it to rainfall thresholds that represent the amount of rain required to cause potential flooding. In the preliminary implementation, we used a

fixed value for the threshold. In the future, we plan to compare the HRRR forecast data against the thresholds provided from the Flash Flood Guidance (FFG) produced by the U.S. National Weather Service (NWS) at the county scale and updated daily based on soil moisture conditions (US National Weather Service, 2017). Currently, the FFGs are published in graphical or CSV format, however NWS plans to make them accessible via webservices. Once the webservices are available, the workflow will be able to access the FFGs automatically each day to have up-to-date thresholds values which are specific to the study area.

A main advantage of the cloud-based approach presented here is that it provides a way to strategically utilize computational resources only when the flood events are likely to occur. Additionally, the workflow is automated, start to finish, without the need for any intermediate human interaction. This means that a decision maker with little or no experience regarding the details of hydrologic modeling, gridded rainfall data, pre- and post-processing procedures, and so forth, can easily execute the workflow and obtain and visualize model results. This work presents a preliminary calibration of the model, but additional work is needed to calibrate and evaluate the model across multiple historical flooding events. It is important to note that this model was so far only tested for Hurricane Sandy. The scarcity of operational river gauges and significant model run-time prior to this research made calibration challenging. The local M2 machine, which was able to run the 15 day Hurricane Sandy model in 2.4 hours, could be used for calibration process. Results of this study suggest a higher resolution grid will improve model accuracy, but this too comes with an increased model run-time. A final challenge that needs more investigation is the differences between CPU and GPU outputs. This difference may become smaller with updates to the model software. TUFLOW plans to release a significantly enhanced version of the GPU model called TUFLOW HPC. This version uses 2nd order solution accuracy solvers rather than the 1st

order that is used in the current version. It will also allow the user to add 2D bridges to the model for better representation within the system and has improvements in the multiple GPU speed performance for executing the model. Finally, more research is needed to see if improving model input data, such as using a finer DEM resolution for portions of the study area or NEXRAD rainfall data, will improve the GPU-based model results.

3.6 References

Brodtkorb, A.R., Sætra, M.L., Altinakar, M., 2012. Efficient Shallow Water Simulations on GPUs: Implementation, Visualization, Verification, and Validation. Computers & Fluids, Vol. 55, pp. 1-12.

Burnash, R., 1995. The NWS river forecasting system catchment modelling. In: Singh, V. (Ed.), Computer Models of Watershed Hydrology. Water Resources Publications, New York, USA, pp. 311e366.

Castro, M.J., Ortega, S., de la Asunción, M., Mantas, J.M., and Gallardo, J.M., 2011. GPU Computing for Shallow Water Flow Simulation Based on Finite Volume Schemes. Comptes Rendus Mécanique, Vol. 339, No. 2-3, pp. 165-184.

Caviedes-Voullième, D., García-Navarro, P. and Murillo, J., 2012. Influence of mesh structure on 2D full shallow water equations and SCS Curve Number simulation of rainfall/runoff events. Journal of Hydrology, 448, pp.39-59.

CEIWR-HEC., 2009. HEC-DSSVue HEC Data Storage System Visual Utility Engine, User's Manual (Computer Program Documentation No. CPD-79). Davis, CA: US Army Corps of Engineers Institute for Water Resources.

Committee on Climate Change and U.S. Transportation, 2008. Potential Impacts of Climate Change on U.S. Transportation. Transportation Research Board Special Report 290. http://www.trb.org/Main/Public/Blurbs/156825.aspx. Accessed September 30, 2015.

David, C.H., Yang, Z.L., Hong, S., 2013. Regional-scale river flow modeling using off-the-shelf runoff products, thousands of mapped rivers and hundreds of stream flow gauges. Environmental modelling & software, 42, pp.116-132.

David, C.H., Maidment, D.R., Niu, G.Y., Yang, Z.L., Habets, F., Eijkhout, V., 2011. River network routing on the NHDPlus dataset. Journal of Hydrometeorology, 12(5), pp.913-934.

Ercan, M.B., Goodall, J.L., Castronova, A.M., Humphrey, M., Beekwilder, N., 2014. Calibration of SWAT models using the cloud. Environmental Modelling & Software, 62, pp.188-196.

Fowler, K.K., 2016. Flood-inundation maps for the Wabash River at New Harmony, Indiana: U.S. Geological Survey Scientific Investigations Report 2016-5119, 14 p.

Garcia, R., Restrepo, P., DeWeese, M., Ziemer, M., Palmer, J., Thornburg J., Lacasta, A., 2015. Advanced GPU Paralellization for Two-Dimensional Operational River Flood Forecasting. In 36th International Association for Hydro-Environment Engineering and Research World Congress. The Hague, Netherlands.

Granell, C., Havlik, D., Schade, S., Sabeur, Z., Delaney, C., Pielorz, J., Usländer, T., Mazzetti, P., Schleidt, K., Kobernus, M., Havlik, F., 2016. Future Internet technologies for environmental applications. Environmental Modelling & Software, 78, pp.1-15.

Grijssen, J., Snoeker, X., Vermeulen, C., El Amin, M., Nur, M., Mohamed, Y., 1992. An information system for flood early warning. In: Saul, A. (Ed.), Floods and Flood Management. Kluwer Academic Publishing, pp. 263-289.

Hassan Water Resources PLC, 2012. Past performance accessed June 12, 2017. http://hwrgov.com/past-performance/.

Hu, Y., Cai, X., DuPont, B., 2015. Design of a web-based application of the coupled multi-agent system model and environmental model for watershed management analysis using Hadoop. Environmental Modelling & Software, 70, pp.149-162.

Huxley, C., Syme, B., 2016. TUFLOW GPU-Best Practice Advice for Hydrologic and Hydraulic Model Simulations. In Proceedings of the 37th Hydrology and Water Resources Symposium (HWRS), Queenstown, New Zealand.

Jacobsen, D., Thibault, J.C., Senocak, I., 2010. An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters. In American Institute of Aeronautics and Astronautics (AIAA) 48th Aerospace Science Meeting Proceedings. Orlando, Florida, USA.

Kalyanapu, A.J., Shankar, S., Pardyjak, E.R., Judi, D.R., Burian, S.J., 2011. Assessment of GPU Computational Enhancement to a 2D Flood Model. Environmental Modelling & Software, Vol. 26, No. 8, pp. 1009-1016.

Kurtz, W., Lapin, A., Schilling, O.S., Tang, Q., Schiller, E., Braun, T., Hunkeler, D., Vereecken, H., Sudicky, E., Kropf, P., Franssen, H.J.H., 2017. Integrating hydrological modelling, data assimilation and cloud computing for real-time management of water resources. Environmental modelling & software, 93, pp.418-435.

Lacasta, A., García-Navarro, P., Burguete, J., Murillo, J., 2013. Preprocess Static Subdomain Decomposition in Practical Cases of 2D Unsteady Hydraulic Simulation. Computers & Fluids, Vol. 80, pp. 225-232.

LeVeque, R.J., 2002. Finite volume methods for hyperbolic problems (Vol. 31). Cambridge university press.

Maidment, D.R., 2016. Conceptual Framework for the National Flood Interoperability Experiment. JAWRA Journal of the American Water Resources Association.

Melillo, J.M., Richmond, T.T., Yohe, G.W., 2014. Climate change impacts in the United States. Third National Climate Assessment.

Moore, R., Jones, D., Bird, P., Cottingham, M., 1990. A basin-wide flow forecasting system for real time flood warning, river control and water management. In: White,W. (Ed.), International Conference on River Flood Hydraulics. John Wiley & Sons, Oxford, UK, pp. 21-30.

National Oceanic and Atmospheric Administration, 2012. Rapid Refresh accessed January 24, 2017. https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/rapid-refresh-rap.

National Oceanic and Atmospheric Administration, 2017a. Advanced Data Access Methods accessed January 24, 2017. http://nomads.ncdc.noaa.gov/guide/?name=advanced.

National Oceanic and Atmospheric Administration, 2017b. Data Products accessed January 24, 2017. https://www.ncdc.noaa.gov/nomads/data-products.

National Research Council, 2009. Mapping the zone: improving flood map accuracy. National Academies Press.

NFIP Statistics, 2016. NFIP Statistics. The official site of the NFIP accessed December 8, 2016. https://www.floodsmart.gov/floodsmart/pages/media_resources/stats.jsp.

Perry,C. A., 2000. Significant Floods in the United States During the 20th Century - USGS Measures a Century of Floods. Kansas Water Science Center accessed December 8, 2016. https://ks.water.usgs.gov/pubs/fact-sheets/fs.024-00.html.

Rostrup, S., Sterck, H.D., 2010. Parallel Hyperbolic PDE Simulation on Clusters: Cell Versus GPU. Computer Physics Communications, Vol. 181, No. 12, p. 2164.

US National Weather Service, 2017. NWS Southern Region Headquarters accessed May 24, 2017. http://www.srh.noaa.gov/rfcshare/ffg.php.

Sanders, B.F., Schubert, J.E., Detwiler, R.L., 2010. ParBreZo: A Parallel, Unstructured Grid, Godunov-Type, Shallow-Water Code for High-Resolution Flood Inundation Modeling at

the Regional Scale. Advances in Water Resources,Vol. 33, Issue 12, December, pp. 1456-1467.

Steissberg, T. E., McPherson, M. M., 2011. HEC-GridUtil Grid Utility Program Managing Gridded Data with HEC-DSS, User's Manual Version 2.0 (Computer Program Documentation No. CPD-89). Davis, CA: US Army Corps of Engineers Institute for Water Resources Hydrologic Engineering Center.

Sun, A., 2013. Enabling collaborative decision-making in watershed management using cloud-computing services. Environmental Modelling & Software, 41, pp.93-97.

Syme, W.J., 2001. TUFLOW-Two & One dimensional unsteady flow Software for rivers, estuaries and coastal waters. In IEAust Water Panel Seminar and Workshop on 2d Flood Modelling, Sydney.

Vacondio, R., Dal Palù, A., Mignosa, P., 2014. GPU-enhanced Finite Volume Shallow Water Solver for Fast Flood Simulations. Environmental Modelling & Software, Vol. 57, pp. 60-75.

BMT WBM, 2016. TUFLOW User Manual. Build 2016-03-AA accessed June 6, 2017. http://www.tuflow.com/Download/TUFLOW/Releases/2016-03/AA/Doc/TUFLOW%20Manual.2016-03-AA.pdf.

Wan, Z., Hong, Y., Khan, S., Gourley, J., Flamig, Z., Kirschbaum, D., Tang, G., 2014. A cloud-based global flood disaster community cyber-infrastructure: Development and demonstration. Environmental Modelling & Software, 58, pp.86-94.

Werner, M., Schellekens, J., Gijsbers, P., van Dijk, M., van den Akker, O., Heynert, K., 2013. The Delft-FEWS flow forecasting system. Environmental Modelling & Software, 40, pp.65-77.

Chapter 4: Design of a Metadata Framework for Environmental Models with an Example Hydrologic Application in HydroShare[3]

4.1 Introduction

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

One way to begin to address these challenges is through better approaches for sharing and reusing models built by others. Just as there has been a major push to make better use of data

[3]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:
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. Environmental Modelling & Software, 93, pp.13-28.
https://doi.org/10.1016/j.envsoft.2017.02.028

collected and maintained by others, the scientific community can benefit from a similar push to make better use of models built by others. Data sharing and reuse has been strengthened through the adoption of agreed-on metadata frameworks. Geospatial data, in particular, has benefited from widely used metadata frameworks that allow scientists and engineers to more easily reuse data collected by others (e.g., ISO, 2003, 2011). More recently, hydrologic time series data have also benefited from the adoption of commonly used metadata frameworks (e.g., Taylor et al., 2014). While many metadata frameworks exist, none specifically addresses computational environmental models. Thus, the objective of this research was to design and implement such a metadata framework for environmental models.

Designing a metadata framework for environmental models poses unique challenges compared to other data types. First, the data required for models are heterogeneous and, in the case of environmental models, input for a single simulation can include dozens, if not hundreds, of data files. These files describe properties of the modeling elements, parameters, forcing functions, boundary conditions, and other data needed to execute the model for a given system. Each model largely adopts its own structure and semantics for storing data, making it difficult to standardize across models. Second, environmental modelers make use of a large and diverse set of computational models; Singh and Woolhiser (2002) cataloged over 65 models focusing on watershed hydrology alone. Environmental modelers will likely continue to make use of a broad range of models because each model is tailored for a given application. As a result, each model adopts unique data structures and semantics for both input and output data. A model metadata framework, therefore, must not force all models into a fixed structure, but rather be flexible and able to accommodate this diversity of models.

Some studies have begun to address the problem of designing a metadata framework for computational models. The Content Standard for Computational Models (Hill et al., 2001)was one of the first attempts at providing detailed metadata about a numerical model that includes the input and output data for model scenarios. Wosniok and Lehfeldt (2013) provide a concept for metadata driven architecture for computational fluid dynamics simulations and a way to integrate model descriptions into spatial data infrastructures. The Community Surface Dynamics Modeling System (CSDMS) created a metadata framework and used it to describe over 180 geoscience models, including over 50 hydrologic models within its model catalog (see http://csdms.colorado.edu). The CSDMS model category focuses on the software for executing a model, what we refer to in this paper as a model program. It does not extend to the input files for a specific model simulation, or what we refer to in this paper as a model instance. The metadata included in CSDMS also do not follow higher-level metadata standards like Dublin Core.

Much of the past research on model metadata has focused on component-based modeling systems. Component-based modeling systems are a tool for integrated environmental modeling where model applications are constructed from a set of "plug-and-play" model components that can be interchanged for different applications (Argent, 2004; Laniak et al., 2013). Metadata frameworks have been proposed for model components generally (Elag and Goodall, 2013), the component interfaces (Gregersen et al., 2007; Peckham et al., 2013), and the variables passed between linked components (Peckham, 2014). Recently, Harpham and Danovaro (2015) designed an un-encoded metadata framework supporting the description of environmental numerical models giving more attention to the construction of model compositions by interfacing model components. This metadata framework was designed to facilitate the description and communication between loosely coupled components of a larger model chain. The framework enables the output from one

model component (e.g., a meteorological model) to be used as an input for another model component (e.g., a hydrological drainage model). This work used the ISO 19115 metadata standard as a starting point and expanded the spatial characteristics, temporal characteristics, and environmental parameters to enable models to be discovered and reused.

Our work is different in that we focus on standalone model programs instead of component-based modeling systems. Standalone model programs can execute a model simulation and generate output, while a model component requires a modeling framework in order to be executed. Model components can be loosely coupled using a modeling framework with other model components, while a model program does not provide this loose coupling capability. We take this focus because, while the adoption of component-based modeling systems is growing, we believe that the vast majority of ongoing studies are using standalone model applications and a metadata framework is needed to enhance the sharing of these standalone model instances. Also, this work could later be merged with past work on model component metadata to create an overarching model metadata framework.

A motivating factor for this research is the design and development of a new system called HydroShare (https://www.hydroshare.org). The goal of HydroShare is to advance hydrologic science by enabling the scientific community to more easily and freely share products resulting from their research - not just the scientific publication summarizing a study, but also the data and models used to create the scientific publication (Horsburgh et al., 2015; Tarboton et al., 2014a, 2014b). HydroShare is a web-based collaborative system developed with the goal of sharing, accessing, and discovering hydrologic data and models (Tarboton et al., 2014a, 2014b). It was designed and built by the authors, along with a larger team of researchers, in collaboration with the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI).

The basic unit of digital content in HydroShare is called a "resource." One of the key steps in designing HydroShare was defining metadata for different resource types (Horsburgh et al., 2015; Tarboton et al., 2014a, 2014b). While users can upload any digital content as a "generic resource" within HydroShare, these generic resources only support basic metadata elements defined by the Dublin Core metadata framework that are applicable to any data type. Specific resource types in HydroShare can extend this Dublin Core metadata to provide new metadata elements that support functionality specific to common hydrologic datasets (Horsburgh et al., 2015). For example, the time series resource types support additional metadata elements relevant to a time series, and the system can automatically plot time series resources because of this metadata (Sadler et al., 2015). Because a model metadata framework like this did not exist for environmental models, we first had to design one. Then, we used the model metadata framework we designed in HydroShare to implement new resource types specific to the needs of environmental models. While the HydroShare implementation motivated the design of the model metadata framework, it is important to emphasize that the metadata framework described here is general and can be adopted for environment models more broadly.

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

4.2 Methodology

4.2.1 Metadata Framework Design

The metadata framework design considers a computational model as two distinct concepts: 1) a model program resource, which includes software for executing a model simulation and generating outputs, and 2) a model instance resource, which includes the input files and, optionally, the output files for a specific simulation. Having model programs and instances as separate resources allows a specific version of a program to be linked to several instances. If model programs and instances were stored together as one resource, the same model program would be stored with each model instance executed by that model program. Additionally, with instances and programs combined, the metadata describing the model program would be repeated with each model instance. This would result in redundant data about the same model program that would need to be entered every time the user uploads an instance for sharing. This may lead to opportunities for inconsistent metadata entry by users for the same model program included in multiple resources. In order to avoid redundantly storing the same program and its metadata with each related model instance, we separated model programs and instances as distinct resource types and implemented an "ExecutedBy" relation as a many-to-one to link between any number of instances and the program used for execution.

The Resource Description Framework (RDF) is used for formally encoding concepts and their associated metadata using a subject, predicate, and object structure (http://www.w3.org/RDF). As a simple example, this basic structure can be used to show that a model instance (subject) is executed by (predicate) a model program (object) (Figure 4.1). Each resource has core metadata defined by the Dublin Core metadata framework and extended metadata designed through this research that is encoded and stored on disk using RDF-XML.

Details of the metadata for model programs and model instances are described in the following subsections.

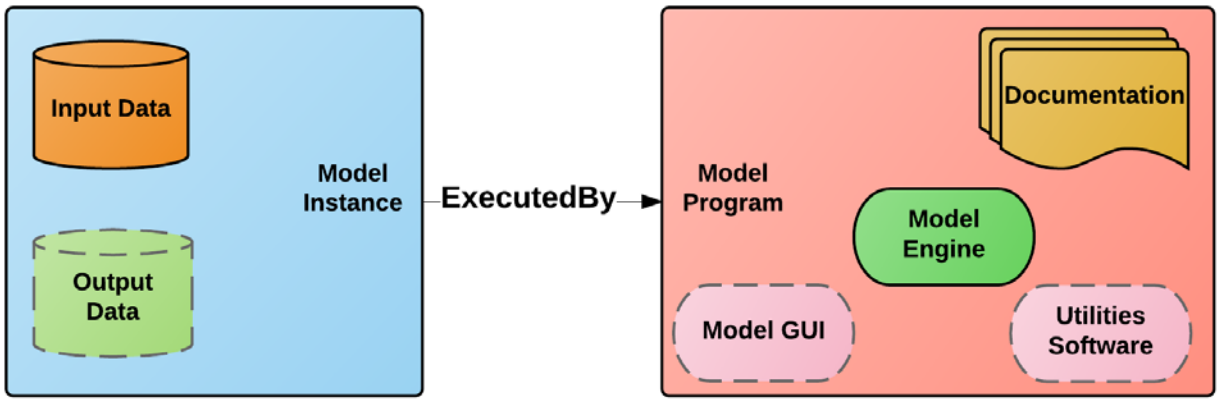


Figure 4.1 Key components of the model program and model instance resources.

4.2.1.1 Model Program Resource Metadata

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

The goal when identifying metadata for a model program was to sufficiently describe a specific version of the software, its computer system compatibility, and its proper and intended use. To foster interoperability, this metadata consists of a basic description of the resource using elements from the Dublin Core metadata standard (shown in Figure 4.2 using the "dc" and "dcterms" prefixes). The basic Dublin Core metadata framework is then extended with resource specific metadata (Figure 4.2; Table 4.1). These extended metadata elements are given names with the "hsterms" prefix, indicating that their names belong to a namespace of terms defined by HydroShare, and are subdivided into content-related and resource-related categories. Content-related metadata includes items such as modelEngine, modelSoftware, modelReleaseNotes, and modelDocumentation to describe the content that should accompany a model program resource. A model program is required to include a model engine, while the other content-related metadata items are optional.

The resource-related metadata describe characteristics of a model program using high-level terminology with the aim of clearly defining and distinguishing between similar model program resources. These include modelReleaseDate, modelWebsite, modelVersion, modelProgramLanguage, modelCodeRepository, and modelOperatingSystem metadata. The modelReleaseDate element provides general information about the environmental model to aid in version identification, while the modelWebsite element is intended to provide users additional model-specific information. The remaining elements describe the software attributes and system compatibility of the model program as shown in Table 4.1. The contents of these metadata elements can serve many different uses, including enhanced search and discovery across a large collection of model program resources. They also aim to support reproducibility by capturing the exact model program used to execute a particular model instance. Some of these metadata elements

(e.g., modelOperatingSystem and modelProgramLanguage) could eventually include and benefit from controlled vocabularies.

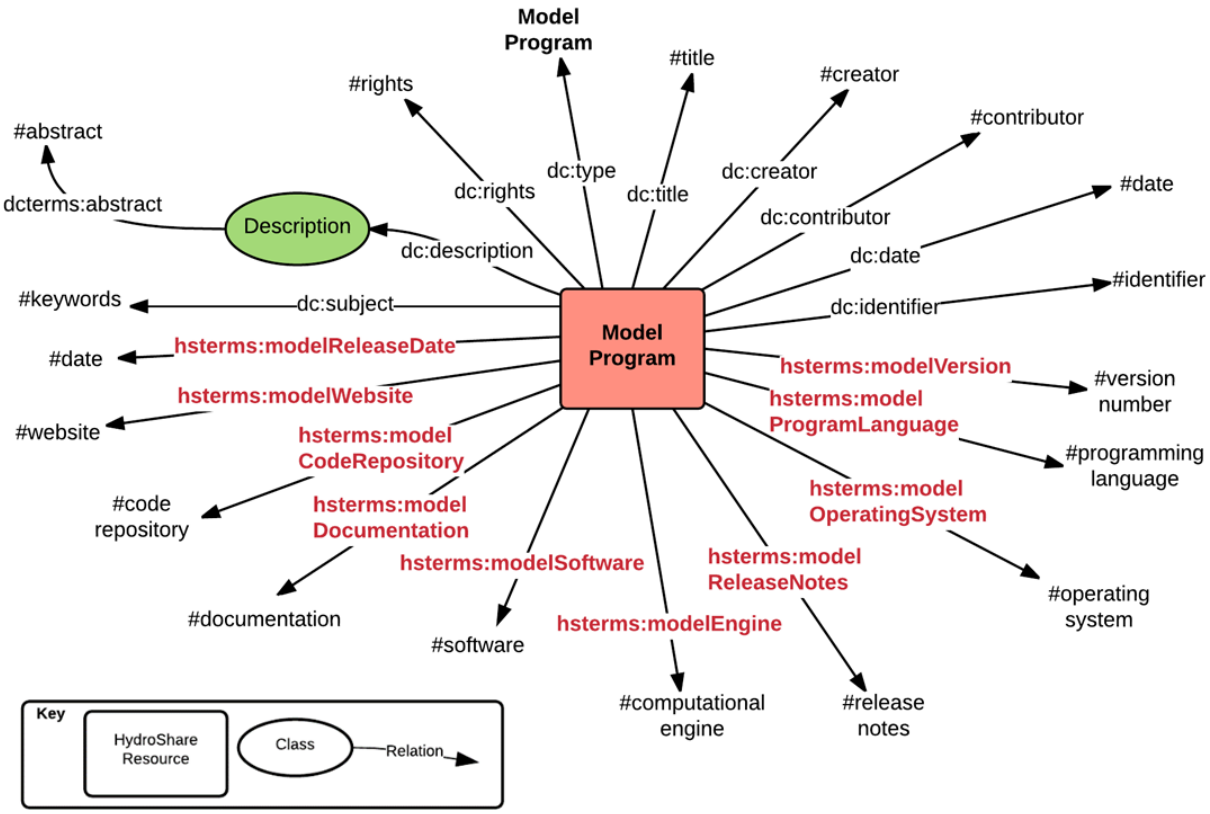


Figure 4.2 Model program resource metadata elements expressed as RDF triples. The # prefix signifies an attribute that can be populated when implementing the metadata framework for a given model program.

Table 4.1 Model program extended metadata element definitions.

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

4.2.1.2 Model Instance Resource Metadata

The model instance resource describes the input files used for execution by a model program. A model instance resource may optionally include the output files resulting after execution. Output for some models can be large. Given that these files can be recreated by executing the model, we made including output files optional. The design for metadata associated with a model instance was intended to capture the aspects required to define and distinguish between different model instances across the wide variety of environmental models. To accomplish this, the design first includes a generic model instance. This generic model instance has metadata elements applicable to any model program instance. The design is extensible including specific model instances that inherit the properties of a generic model instance and add new properties that are relevant to one or more model programs. This pattern is illustrated in Figure 4.3. In this figure, some specific

model instance resources are listed as examples, with the idea that this list can be extended to include other environmental models as well. This design, therefore, provides two ways to capture metadata for a model instance. The default option would be to use a generic model instance resource type. However, if available, a specific model instance resource type should be used to take advantage of enhanced functionality and metadata capture.

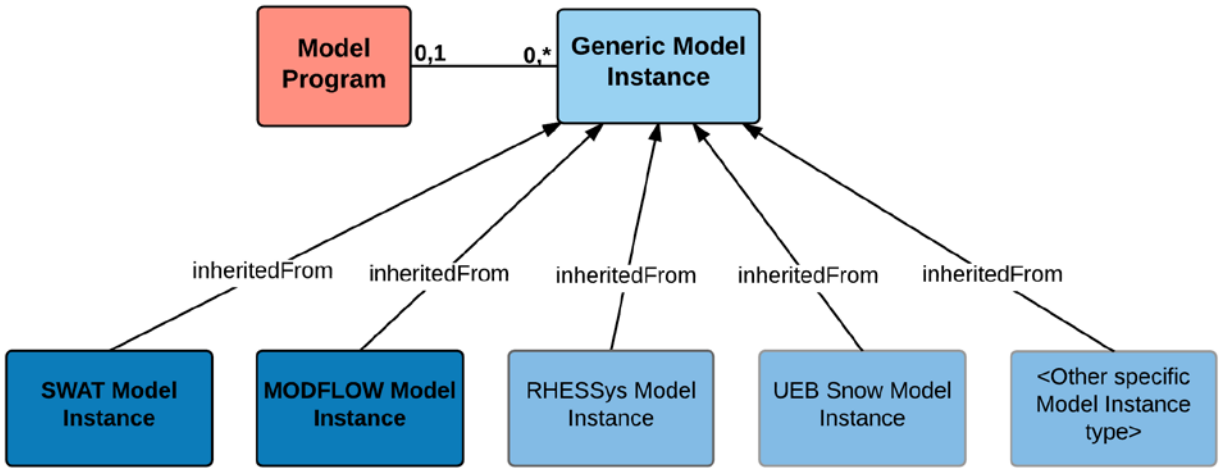


Figure 4.3 Generic model instance and specific model instance hierarchy. Model program, generic model instance, SWAT model instance, and MODFLOW model instance metadata have already been designed, while metadata for the other specific model instances are either in development or planned for the near future.

Figure 4.4 presents the metadata for a generic model instance. Because the generic model instance extends the Dublin Core metadata framework, it inherits the metadata elements defined by Dublin Core (with names shown using the "dc" and "dcterms" prefix). One metadata element defined in Dublin Core that is particularly important for model instances is the coverage element. This metadata element defines the temporal and spatial extent of a resource. For a model instance resource, the temporal coverage provides the start and end date/time for the simulation; the spatial coverage provides a place name and geographic coordinates for the model instance. The spatial coverage can be represented by a point (e.g., the centroid of the modeling domain) or a box (e.g.,

the bounding box of the modeling domain). This coverage element does not represent the exact shape of the model instance, but rather its geographic location or extent.

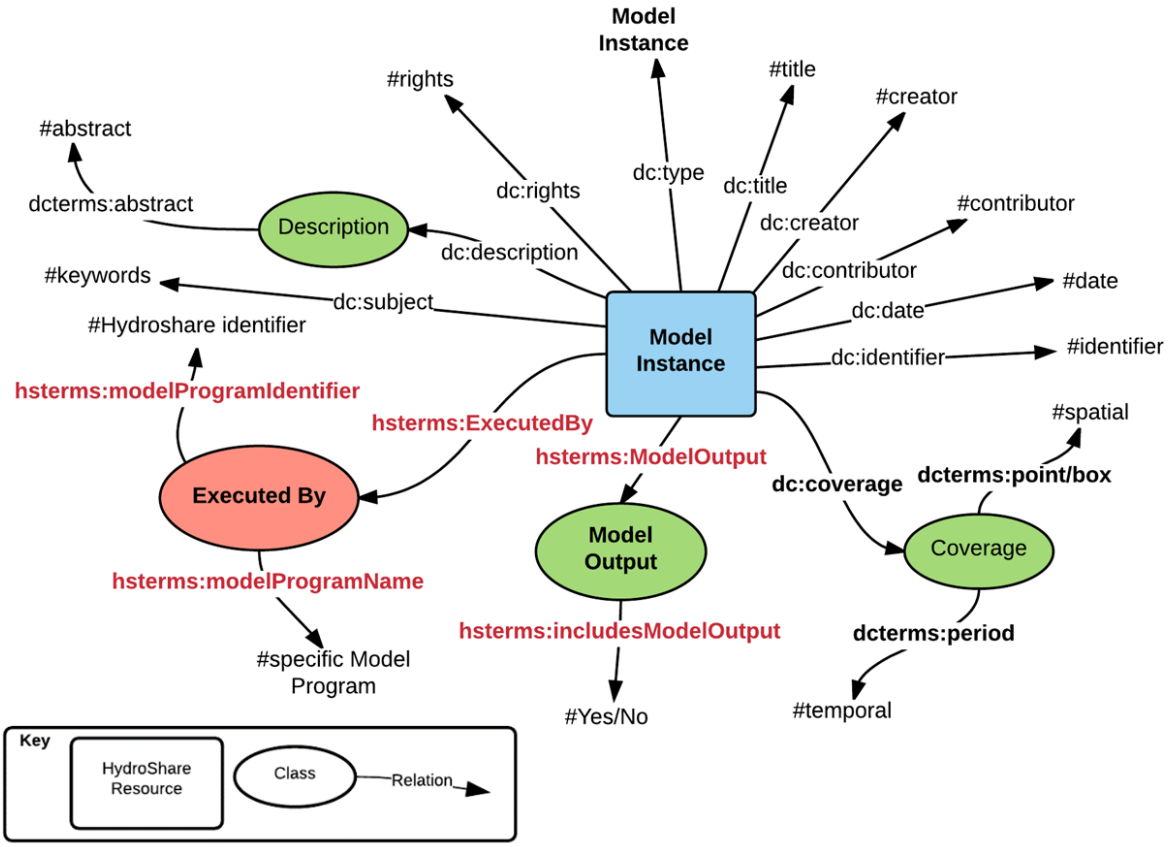


Figure 4.4 Generic model instance resource metadata elements expressed as RDF triples.

Table 4.2 Generic model instance extended metadata element definitions.

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

As with the model program, the generic model instance metadata is extended from the Dublin Core elements with the names of additional metadata elements having the "hsterms" prefix (Figure 4.4; Table 4.2). These metadata elements are subdivided into two main classes: ModelOutput and ExecutedBy. ModelOutput includes information about the output data generated by the model after it is executed. Only one element was deemed necessary in the initial design for describing the model output, although more elements could be added later. The element included is includesModelOutput, which allows users to indicate if the output files are included along with the input files as part of the model instance resource. The ExecutedBy element links the model instance resource with the model program resource that is used for execution. ExecutedBy includes two sub-metadata elements: modelProgramName and modelProgramIdentifier. The modelProgramName element stores the name of the linked model program resource, while modelProgramIdentifier stores its unique identifier. By linking a model instance to a model program resource, the ExecutedBy metadata element facilitates later reproducibility of the model results.

As an example of a specific model instance, consider an extension to the generic model instance designed to add metadata specific to an instance of the Soil and Water Assessment Tool (SWAT). This SWAT model instance offers extended metadata elements that more fully describe SWAT model instances, but that are not directly applicable to other environmental models. The SWAT model instance was designed to be compatible with the SWATShare application, which is an interactive Web tool used to run, visualize, and interact with SWAT model instances (Rajib et al., 2016). The extended metadata elements for a SWAT model instance are shown in Figure 4.5, and the extended metadata elements are defined in Table 4.3. While these elements are specific and extensive, many of them are optional so the barrier to entry is still low. Also, through future work, many of the metadata elements could be extracted automatically from model instance configuration files. Unlike the generic model instance, the SWAT model instance introduces controlled vocabularies for some SWAT model metadata elements including modelObjective, simulationType, and simulationTimeStepType. These controlled vocabularies are compatible with the controlled vocabularies used by SWATShare. For example, simulationType has a controlled vocabulary consisting of three choices: normal simulation, sensitivity analysis, and auto-calibration.

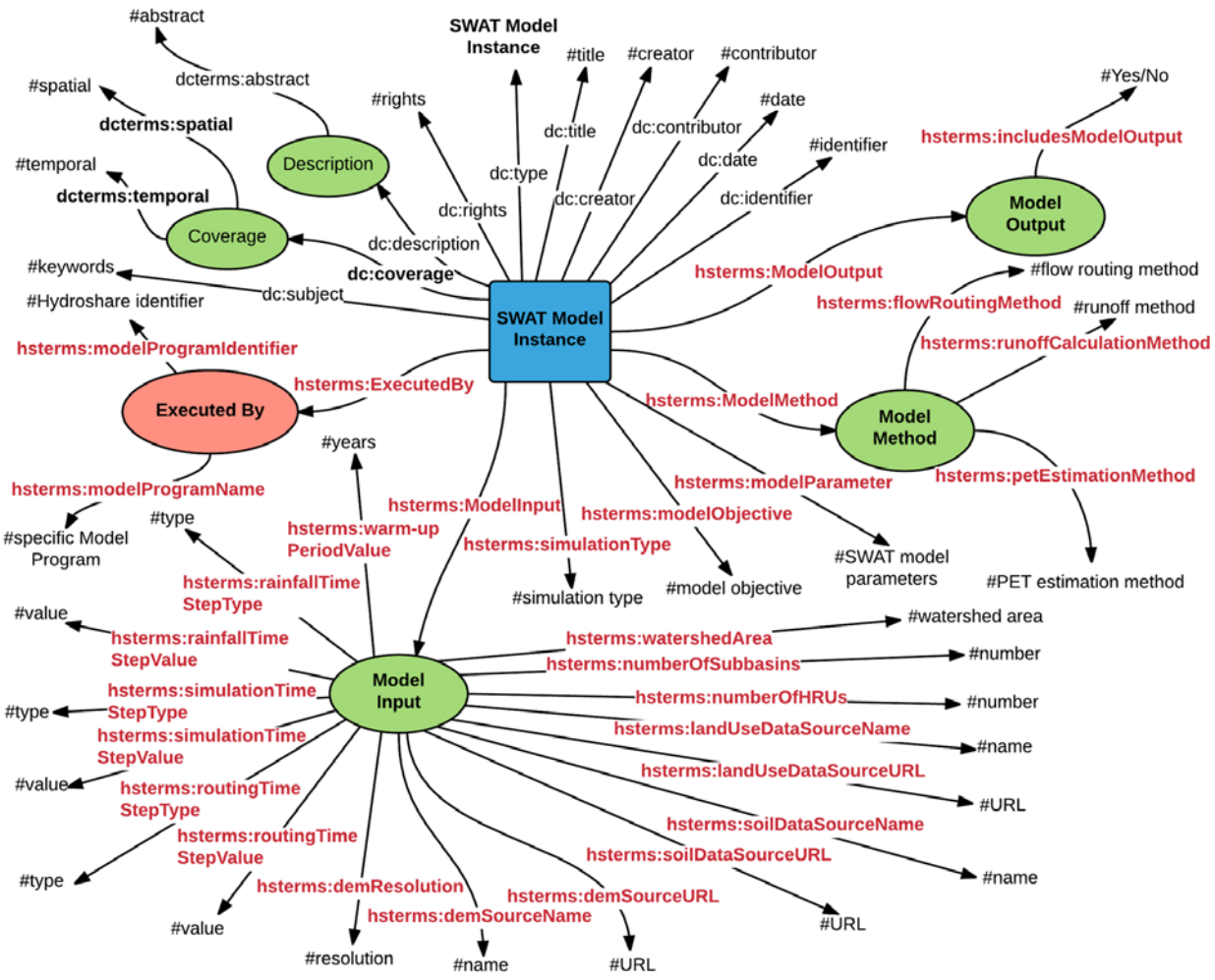


Figure 4.5 SWAT model instance metadata expressed as RDF triples.

Table 4.3 SWAT model instance extended metadata element definitions.

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

While SWAT is used to provide an example of a specific model instance, similar metadata and corresponding controlled vocabularies could be developed for other models. The design goal of this work, however, was not to capture metadata relevant to all environmental models, as doing so would be impractical. Rather, our goal was to design a framework that has a common core and a clear methodology for extending this core for specific environmental models. We plan to provide examples, like the SWAT example, that third party developers can follow to create their own specific model instance metadata. By providing a common foundation for metadata and resource-structure across models, there will be a level of standardization that will aid in interoperability across software systems. Specific model metadata acknowledges the diversity among environmental models and does not force conformity to a single set of metadata elements. The design also allows for changes in the future. For example, if additional common model metadata elements are identified across environmental models, then they can be added to the generic model instance class and inherited by all specific model instances.

4.2.2 Experimental Use Case

To demonstrate the metadata design, we used the application of a SWMM model from Chapter 2 used to study flooding in an urban watershed (Morsy et al., 2016) as a use case. We wish to publish the resulting model instances online. There are many motivating factors for doing this. First, we believe that a model instance, like the journal paper, is an important product from the research and should stand on its own as a citable product. Second, we want to foster ways for other scientists to build from or reuse our model to address their own scientific research questions. Third, we want to ensure that the model program used in our study, including the model engine, utility software, and documentation, is captured within a single online resource. This is important

because, after some time, the model program developers may not provide this particular version of the software on their website. Lastly, this is a way of meeting the research sponsor's data management obligations. While this use case is specific to scientific research, a similar use case could be followed for consulting or industrial modeling activities. While such model applications may not result in journal publications, there is still significant value in descriptive metadata for internal cataloging and archiving purposes. Additionally, in such cases models can be shared privately within HydroShare allowing collaboration among specific users while keeping the data, model, and results confidential.

As a reminder, the objective of this prior modeling study was to better understand the potential of rain gardens as distributed stormwater controls for flood mitigation within an urbanized watershed (Morsy et al., 2016). The specific study area of the research was the Rocky Branch watershed, which is located in downtown Columbia, South Carolina, USA. Because a significant portion of the watershed is developed, high intensity storms that typically occur during the summertime result in flooding at different locations within the watershed. For this study, two different model instances were created (Figure 4.6). The first model instance is a well-calibrated and evaluated model that simulates flooding events in the Rocky Branch watershed. The second model instance builds from the first model instance and includes additional, hypothetical rain gardens as stormwater controls to test if their addition mitigates flooding in the watershed for storm events with different return periods.

The metadata framework was implemented within HydroShare and used to share the model program and model instance resources for the example application. HydroShare, as introduced earlier, is an online system for managing resources adhering to a Resource Data Model (Horsburgh et al., 2015; Tarboton et al., 2014a). The HydroShare architecture organized as shown in Figure

4.7 (Heard et al., 2014) consists of open source components including Django, a web application platform, Mezzanine, a content management system meta-framework, and the Integrated Rule-Oriented Data System (iRODS), an enterprise storage management middleware (Rajasekar et al., 2010). Results detailing the technical aspects of the software implementation are presented in Section 4.3.1.

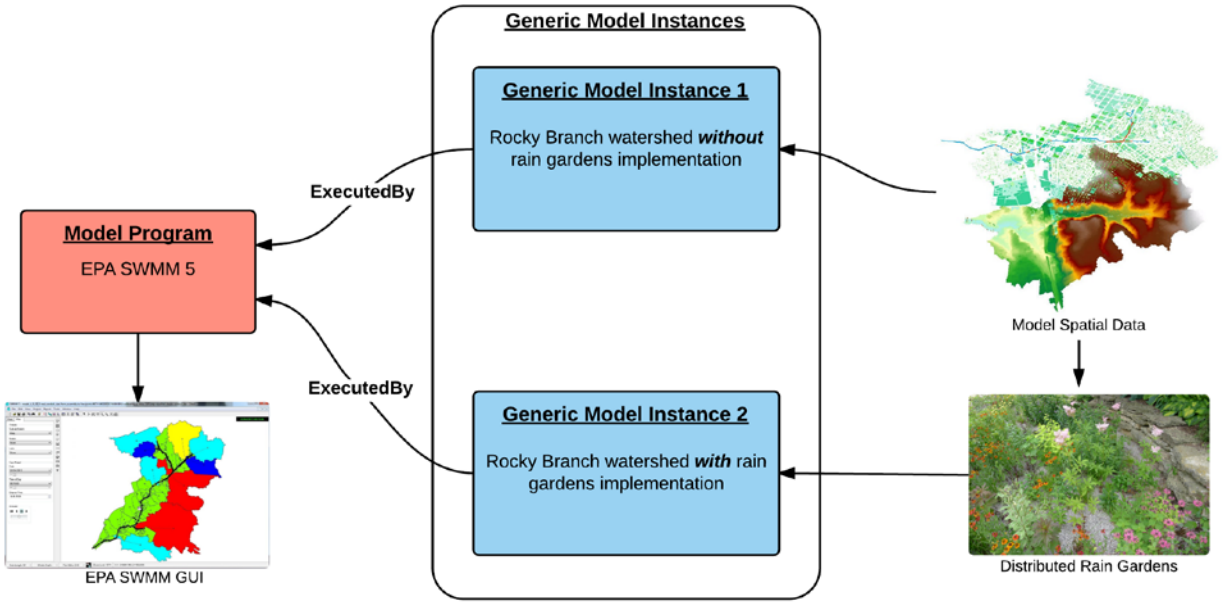


Figure 4.6 Use case implementation as a model program and two model instance resource types.

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

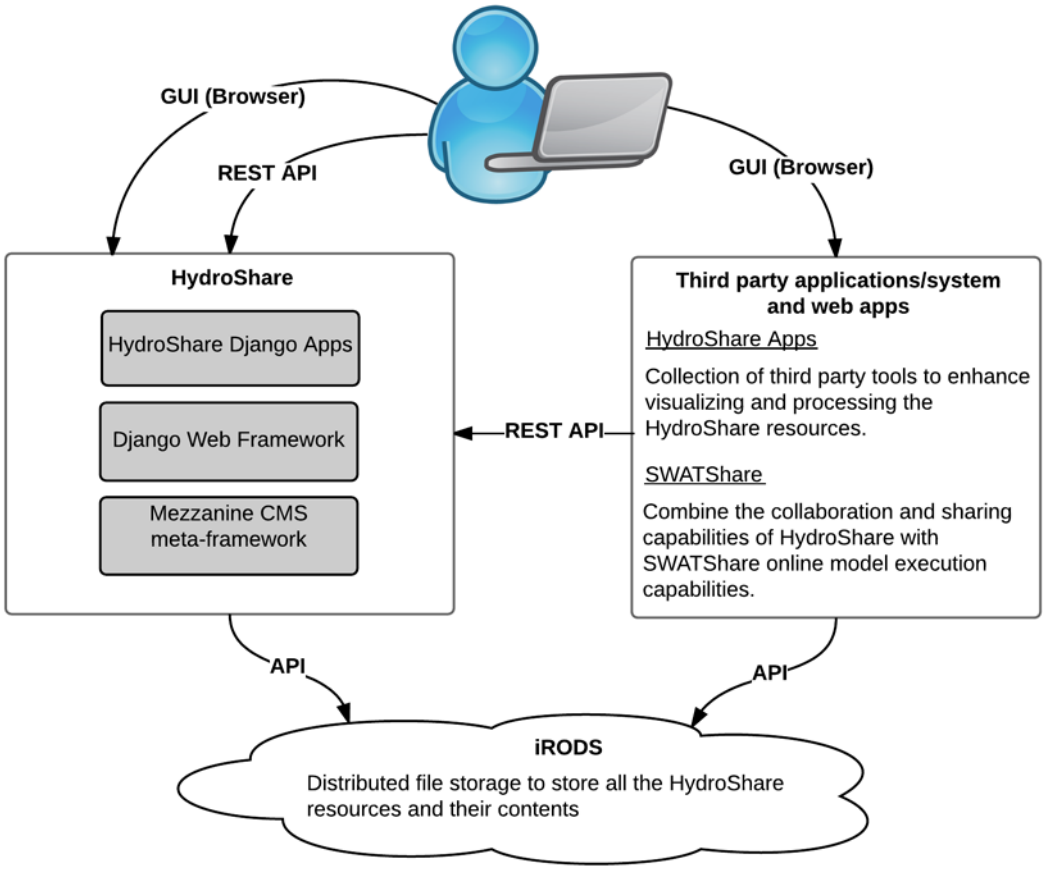


Figure 4.7 HydroShare's general architecture emphasizing the connections between the user, HydroShare, iRODS, and third party applications.

4.3 Results

4.3.1 Results for Software Implementation within HydroShare

Figure 4.8 shows the class structure for the new model resource types implemented within HydroShare based on the metadata framework design. Each resource type consists of three main categories of classes: the resource data type class, the classes for the individual extended metadata elements, and the container class that groups all metadata elements. For example, the classes in the three categories for the model instance resource type are 1) ModelInstanceResource, which is the resource data type class, 2) ModelOutput and ExecutedBy, which are the classes representing

the extended metadata elements, and 3) ModelInstanceMetaData, which is the class that contains all the metadata elements. The resource type classes for model instance and model program inherit from the BaseResource class, which, in turn, inherits from the Abstract Resource class. This structure allows the model resource type to inherit the Dublin Core metadata elements. Specific model instance metadata, like that for the SWAT model instance resource type, inherits from the generic model instance resource type class. The diagram shown in Figure 4.8, therefore, could be extended for other specific model instance metadata.

Each Model resource type extends the BaseResource class by representing specific metadata elements as individual classes. These extended metadata classes inherit from the AbstractMetaDataElement class. In this class, there is one required attribute: term. Other attributes needed for further description can be added. For example, the extended metadata class ExecutedBy for the ModelInstance resource has the model_name, and model_program_fk attributes. The specific metadata elements are grouped in the CoreMetaData class. The ModelProgramMetaData, and ModelInstanceMetaData classes inherit from the CoreMetaData class, which is the metadata container that includes the common metadata element objects. These classes are the link between the ModelProgramResource, the ModelInstanceResource classes, and their extended metadata classes. One-to-one relationships are made between ModelProgramMetaData and ModelInstanceMetaData classes and each of their respective extended metadata classes. These extended metadata classes are then included as supported metadata elements for their related resources (ModelProgram or ModelInstance resources) where they could be used to create, update, and delete class instances associated with these resource types.

An important method of the CoreMetaData, ModelProgramMetaData, ModelInstanceMetaData, and SWATModelInstanceMetaData is get_xml. This method converts

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

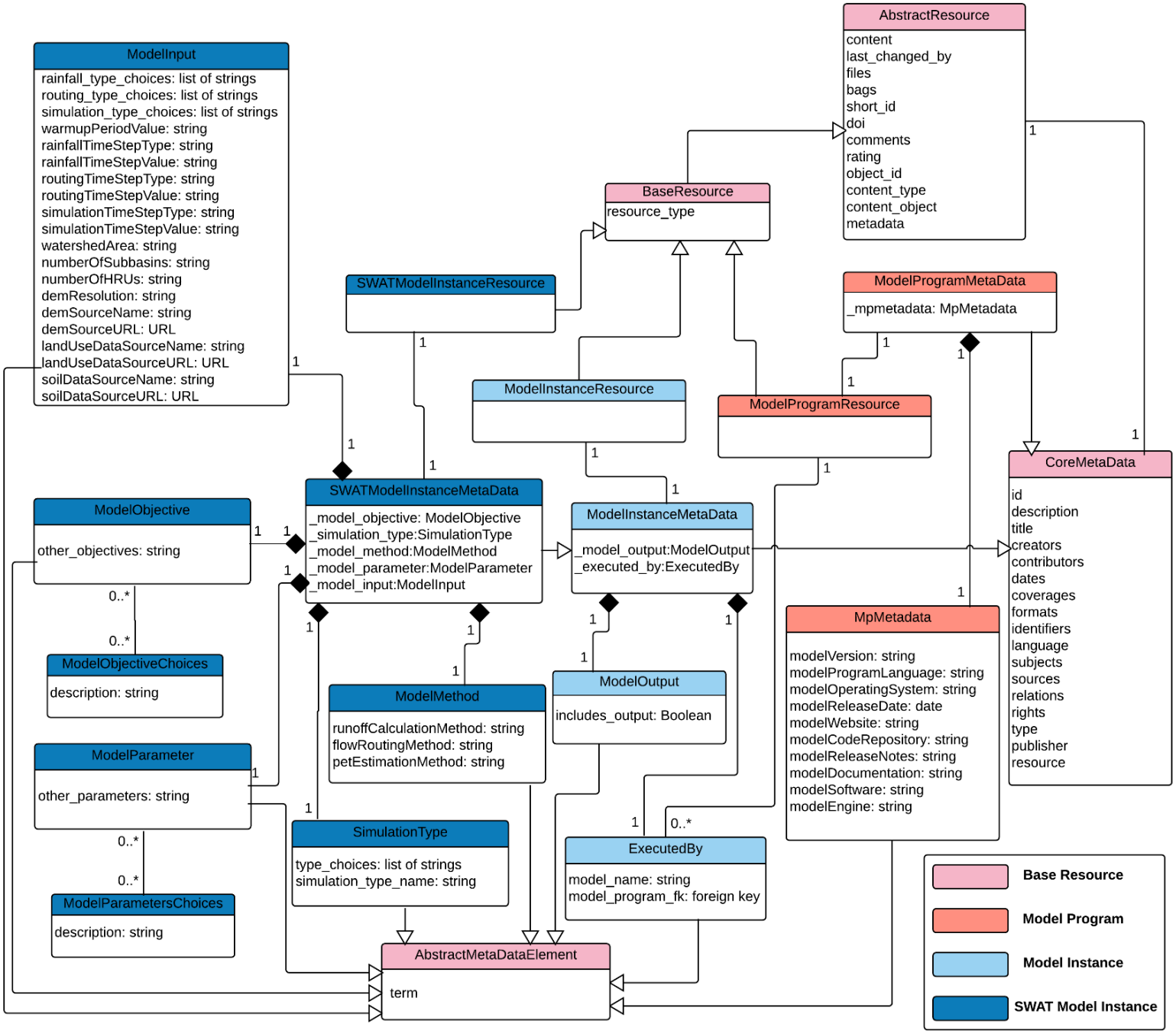


Figure 4.8 Metadata classes for model resources implemented within HydroShare.

4.3.2 Results from the Example Use Case

Figure 4.9 illustrates the metadata that can be captured for the example use case using the generic model instance and model program resources. Each resource has a title, creator, and other metadata that follow the Dublin Core metadata standard. In addition, extended metadata elements for each resource (with names shown using the "hsterms" prefix) help to more fully describe the model instance and corresponding model program used for executing the model instance. Figure 4.9 also shows how the model program resource type, in this case the SWMM model (Rossman et al., 2016), and the model instance resource type, in this case a Rocky Branch watershed simulation, are connected using the ExecutedBy relationship.

Figure 4.10 is an activity diagram showing the steps used to create new model resources on hydroshare.org. Three resources were created in this example: a model program resource for the EPA-SWMM model version 5.1.009 (Rossman et al., 2016) and two model instance resources for the Rocky Branch watershed simulations (e.g., Morsy, 2015). Figure 4.11 shows the Graphical User Interface (GUI) for how a user selects a model resource type within HydroShare. In the current implementation, the model resource types are grouped together under the modeling title. Once the user selects the desired resource type, adds a title, and uploads the related files, the new resource is created in HydroShare and the user sees the landing page for this newly created resource. At this point, a unique identifier specific to the HydroShare system has been automatically assigned to the resource. Later, if the user decides to formally publish the resource in HydroShare, a more formal digital object identifier (DOI) would be assigned to the resource. After a resource is formally published and a DOI is assigned, the user can no longer make changes to the resource metadata or the uploaded files. Prior to formal publication, authorized users can make changes to the resource at any time.

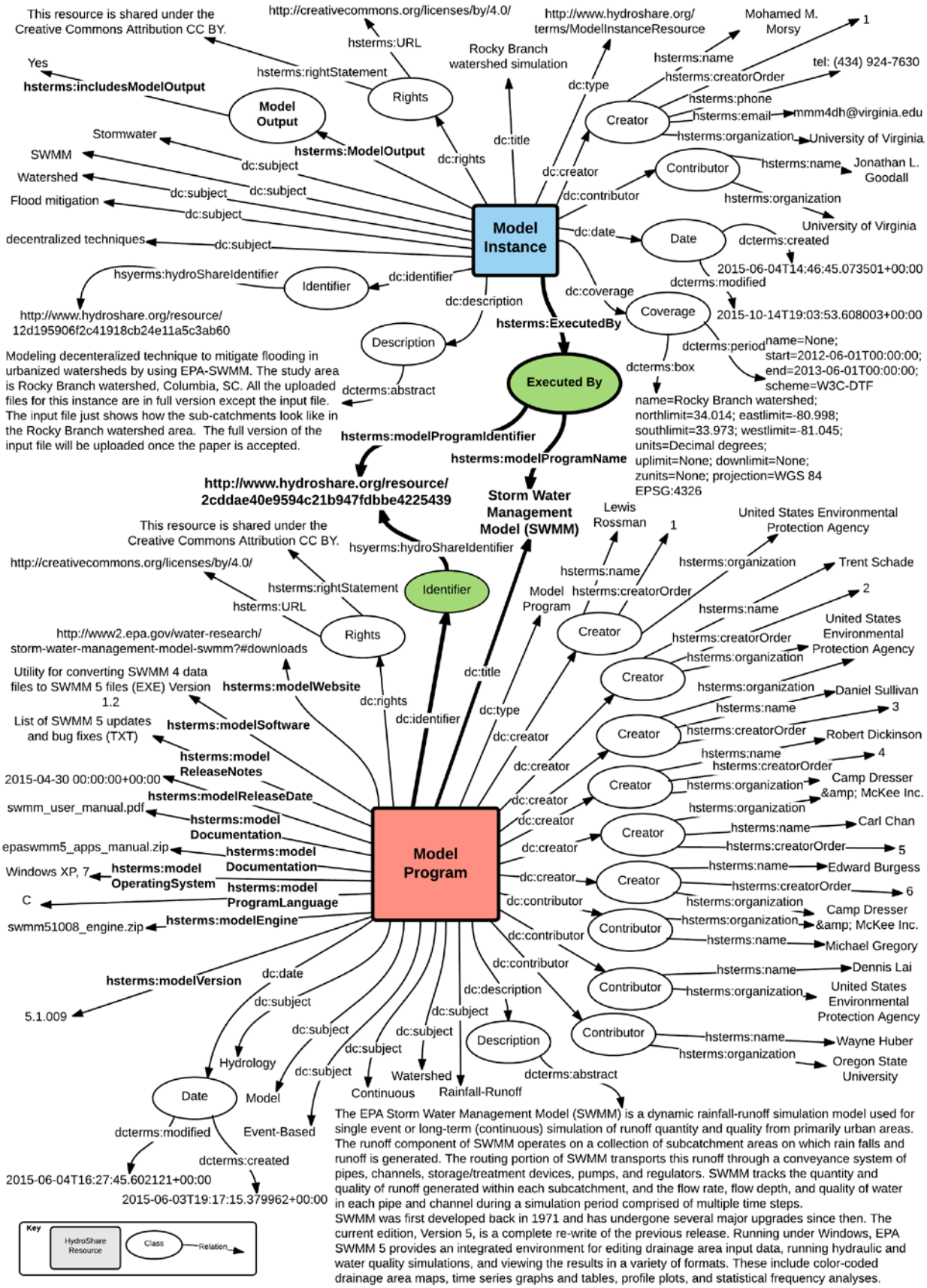


Figure 4.9 Results of populating the model instance and model program metadata for the example use case.

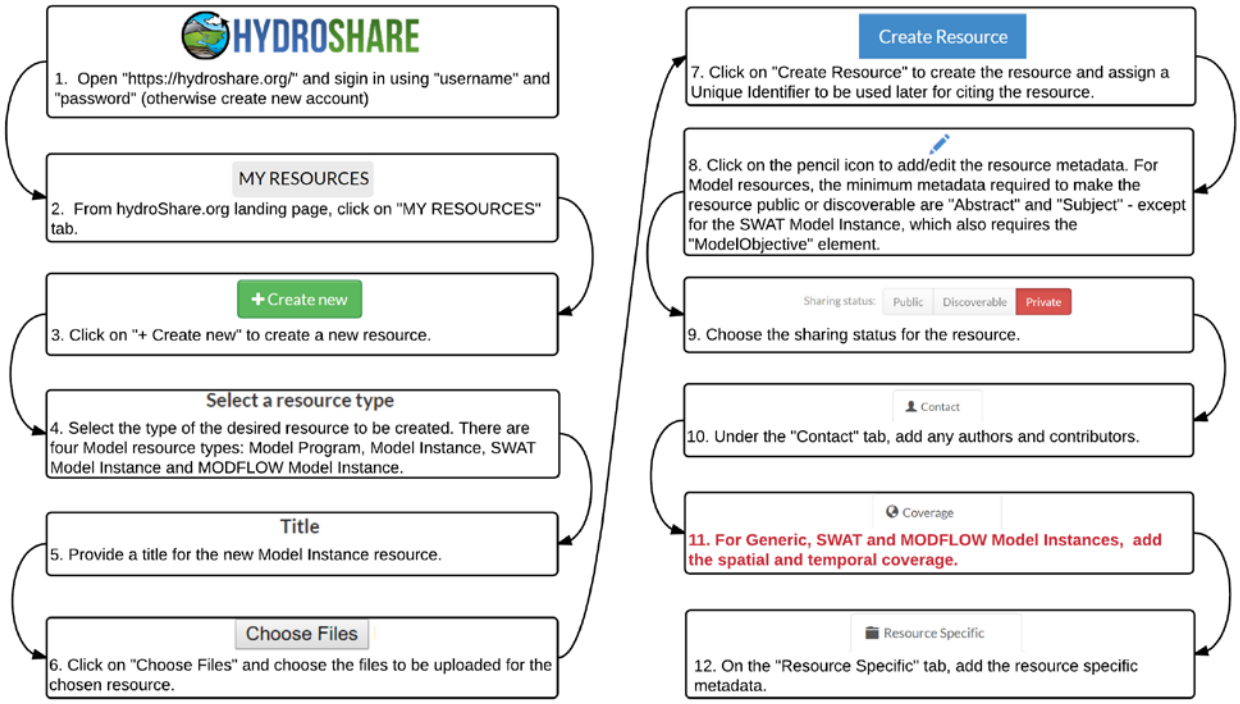


Figure 4.10 Activity diagram describing the steps required to create a new model instance resource within HydroShare. Step 11 is highlighted to indicate that only model instances require coverage and not model programs.

Figures 4.12 and 4.13 show the resource specific metadata for the model program resource and the generic model instance resource types, respectively, on their landing pages in HydroShare. These figures show HydroShare's metadata "edit" mode to illustrate all of the available metadata elements, as HydroShare's default is to hide metadata elements for which there are no values in "view" mode. Note that the model instance is linked to the model program used for execution (Figure 4.13). Under the "Model Program used for execution" heading on the generic model instance landing page, there is a dropdown list that collects all the available public model program resources in HydroShare. The user chooses the model program resource used to execute the model instance resource from the dropdown list (or creates a new model program resource if it is not already available). Once the user chooses the desired model program resource, a summary of the model program metadata is displayed to aid the user in confirming that the correct model program was selected.

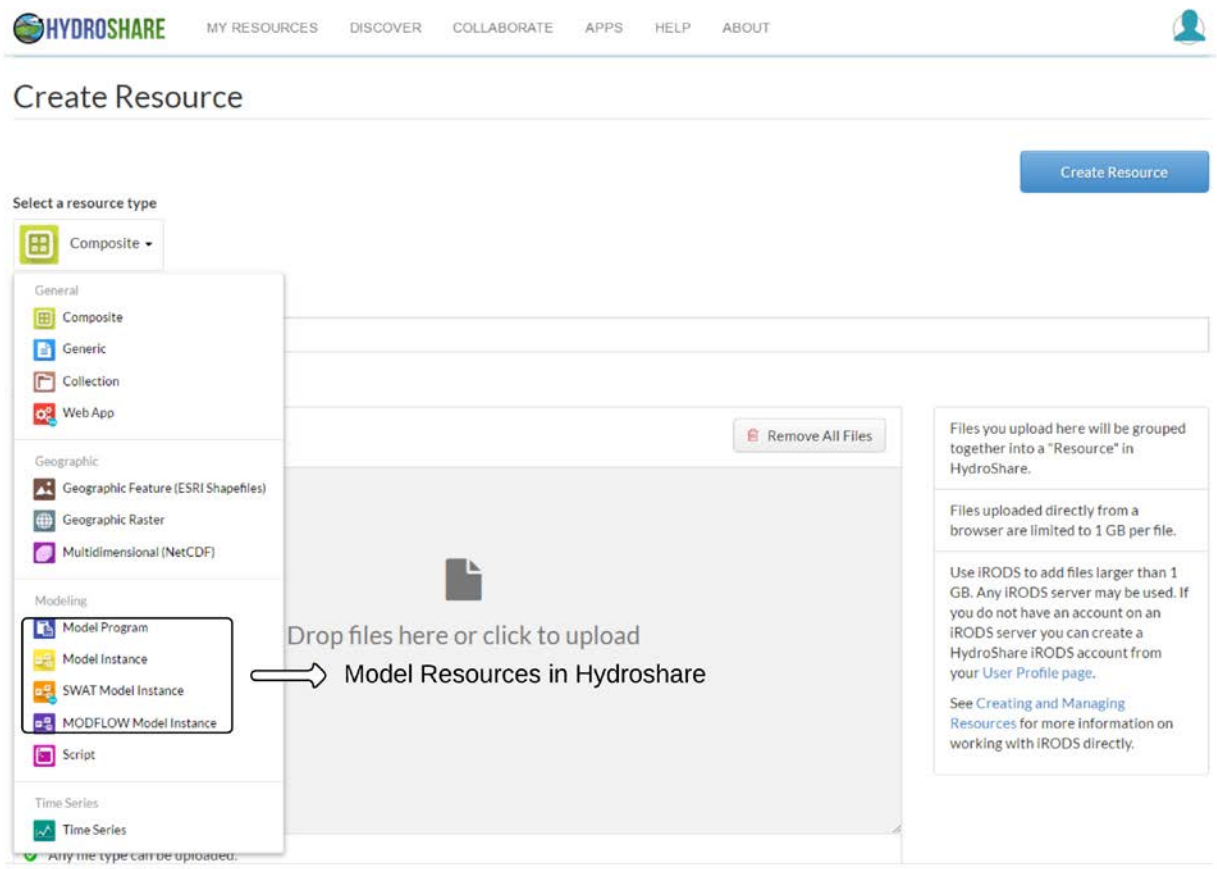


Figure 4.11 Screen shot showing model resource types currently implemented on hydroshare.org.

Another important aspect of the model instance resource is the coverage metadata. Figure 4.14 shows how the coverage metadata appears in the resource's landing page in edit mode. As explained above, there are two types of coverage metadata elements: spatial and temporal. All of the spatial metadata is expressed in World Geodetic System (WGS) 84 coordinates, which is used throughout HydroShare. This allows standard web tools to search the metadata easily without full GIS functionality. However, users must be aware that errors can be introduced if the spatial data is transformed from another coordinate system to WGS 84. For the use case, the spatial metadata was entered for this model instance as a two-dimensional bounding box (rather than an XY point). Once the user inserts the bounding coordinates, the box will appear on the map so that the user can confirm the spatial coverage extent. The user can also specify the coverage by clicking a point on

the map or dragging a box on the map. The temporal coverage metadata consists of start and end dates and times for the model instance. This is implemented in the data model based on the W3C-DTF scheme, which by default enables full specification of a date/time string, including a time zone. Currently, as seen in Figure 4.14, the HydroShare interface supports only the entry of dates without times or time zone specifications. HydroShare uses this coverage metadata to support both spatial (e.g., map-based) and temporal searches to identify relevant resources.

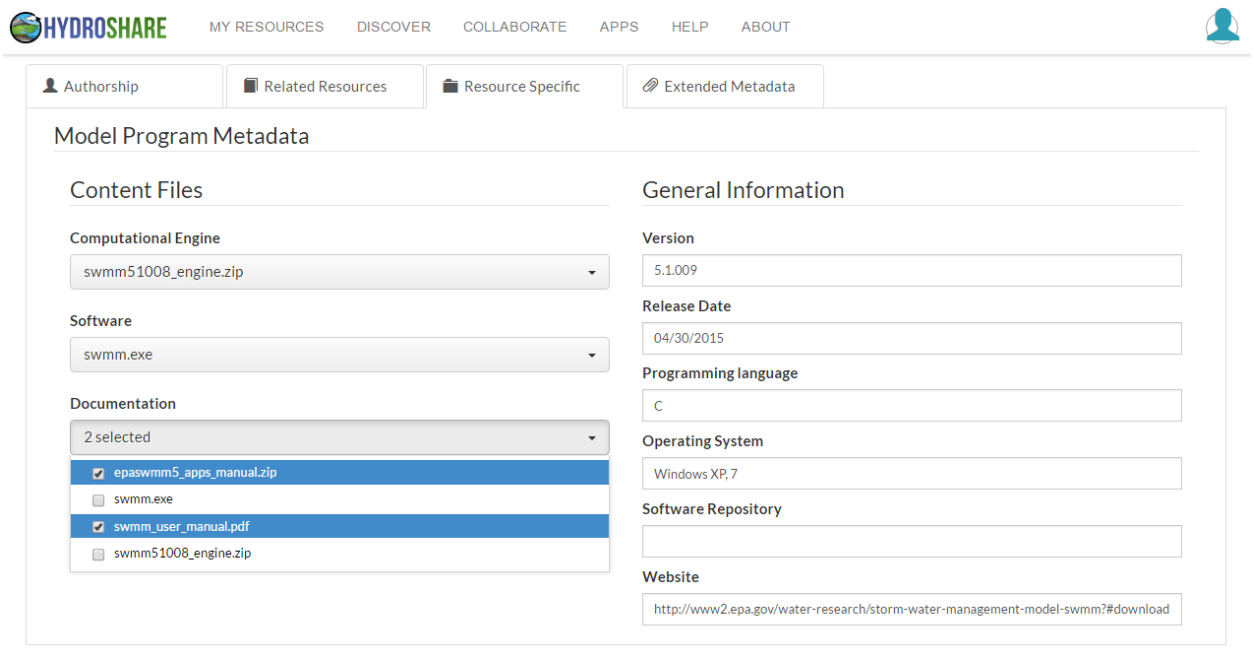


Figure 4.12 Model program resource specific metadata on the resource's landing page on hydroshare.org (shown in edit mode).

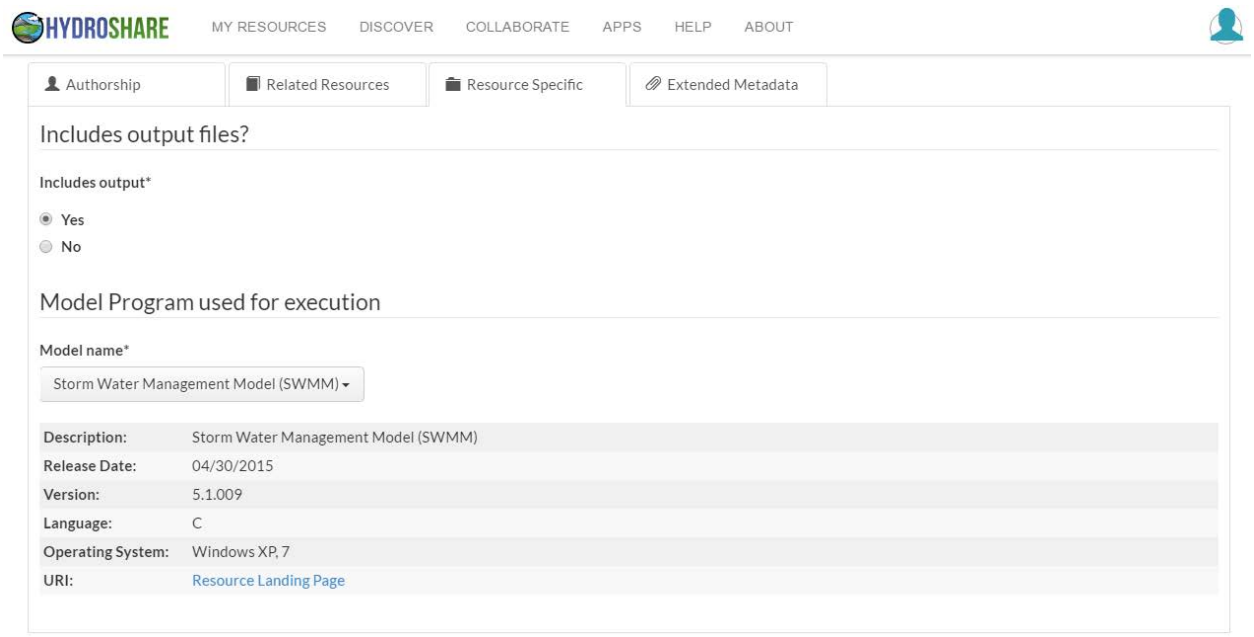


Figure 4.13 Generic model instance resource specific metadata on the resource's landing page on hydroshare.org (shown in edit mode).

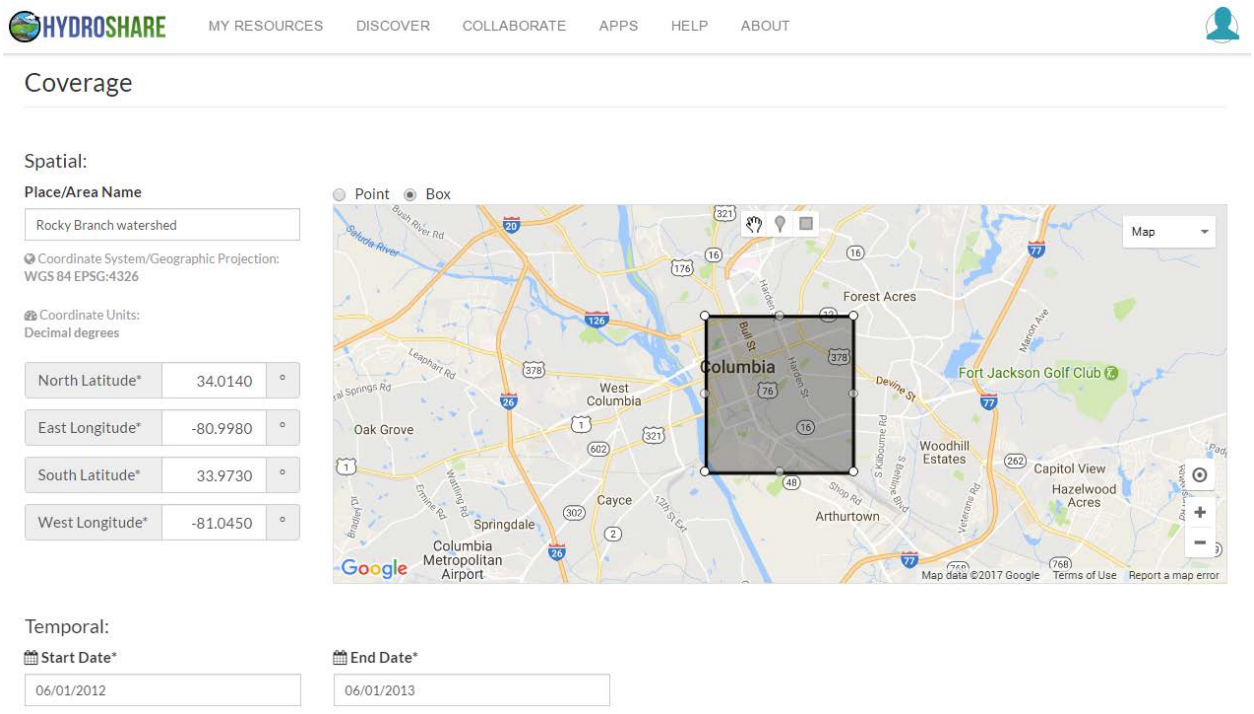


Figure 4.14 Model instance resource type coverage metadata on the resource's landing page (shown in edit mode) on hydroshare.org.

4.4 Discussion

The metadata framework proposed in this study was designed to provide a balance between simplicity and complexity; simplicity to encourage to sharing of models by model producers, and complexity by providing a sufficient level of information to enable discovery and use of the model by potential consumers. One of the most difficult design decisions in this work was to separate model programs and model instances into two different resources rather than a single combined resource. The design decision was made for the following reasons. First, it allows the model program metadata to be entered only once within the system. Second, it simplifies the task of identifying all model instances executed by a given model program stored within the system. Third, it provides a path for online execution of many model instances that are linked to a single model program. We felt these benefits outweighed the added complexity and management needs introduced by separating the model program and model instance concepts into different resources types. We acknowledge that some use cases require incremental changes to a model program's source code, and we are considering options for capturing these incremental changes to model programs without the need to create a completely new resource every time a model program's source code has been changed. That said, users are not restricted from uploading a model program within a model instance, if desired. If this becomes common practice, we are considering allowing a model instance resource's ExecutedBy field to point to itself. This would signify to a user that the model program, whether it be a complied binary file or the source code, is located within the model instance resource.

Another key design decision was to allow a model instance resource to be linked to only one model program resource. We realize that it is possible for a model instance to be executed successfully by multiple model program resources (e.g., two model programs with different

versions but compatible with the same model instance). However, allowing a model instance to be linked to more than one model program would introduce uncertainty about what program was used to execute the instance for a given study. Reproducibility could be compromised as a result because executing the model instance with a different model program may return slightly different results. For this reason, the design requires a model instance to be linked to only one model program.

We encountered through the use case application the important issue of how to handle the case where the person uploading a resource into HydroShare, what HydroShare refers to as the resource's owner, is not the author of that resource. HydroShare separates intellectual credit attribution from access control and management of content. The Dublin Core vocabulary term "Creator" is used in HydroShare metadata for the intellectual originator of the content. This is displayed as Author on landing pages and used in citations. The term "Owner" is used in access control and management of content and is typically the HydroShare user responsible for uploading the content (although ownership can be transferred after uploading, and others can be assigned permissions to edit and upload content). In the SWMM model program resource example, the EPA-SWMM model was authored by researchers at the United States Environmental Protection Agency (EPA) but was uploaded to HydroShare by the modeler, one of the authors of this paper. The original authors of SWMM were entered as authors for the resource and the relationship "isCopiedFrom" was added to the resource pointing to the website from which the model program was obtained. With this added relationship, the HydroShare system automatically generates and displays a citation on the resource's landing page that shows that the resource in HydroShare was replicated from an external source, as shown below. The user that uploaded the resource into HydroShare, but did not author the resource, remains the resource owner but rightly does not receive authorship credit for this resource within the citation.

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

A limitation of this work at its current stage is the ability to scale-up to support dozens of different specific model instance resource types. Ideally, the creation of new HydroShare resource types would be simple enough that it could be done by the broader community of model developers. Currently, however, the process of creating a new resource type within HydroShare is time consuming and requires advanced knowledge of the HydroShare system and architecture. One approach to address this would be to focus on simplifying the process for creating new resource types. Another possibility would be to alter the approach described in this paper so that specific model instances are not implemented as new resource types, but still can have extended metadata for specific model programs. In this case, all model instances would be uploaded using a single resource type, but there would be a mechanism to filter the metadata fields available to the user once the user or system identifies the uploaded model instance as being a specific and known type (e.g., a SWAT model instance). More research is needed to test these alternative options in terms of their practicality, usability, and scalability within HydroShare.

Discovery is an important use case that model metadata must support. In HydroShare, the metadata model for all resources was designed to support discovery. However, the search interface design that exposes metadata elements within the existing data model is still under active development. Currently, users can discover HydroShare resources by searching and filtering model resources using many of the Dublin Core metadata elements implemented in the HydroShare data model (i.e., the generic resource metadata). For example, resources can be discovered by model authors (dc:creator), model resource type (dc:type), model keywords (dc:subject), full text search

of a model resource description (dc:description/dcterms:abstract), model spatial location (dc:coverage/dcterms:box or dcterms:point), and model temporal duration (dc:coverage/dcterms:period). Other Dublin core elements (e.g., dc:language) have not yet been exposed as discovery facets.

The HydroShare system does not yet allow for discovering resources using the specific metadata designed for each resource type. Using the resource specific metadata defined for model instances and programs in this research, however, will further enhance and improve the discovery capabilities. For example, if a user would like to discover all model instance resources within HydroShare that include output files along with model input files, the system could use the metadata element ModelOutput/includesModelOutput. If a user would like to discover all the model instance resource types that are executed by a specific model program available in the HydroShare system, the system could use the metadata element ExecutedBy/modelProgramName. Also, if a user would like to discover all model program resources that are compatible with a specific operating system, the system could use the metadata element modelOperatingSystem.

As HydroShare continuities to evolve, the types of searches users wish to complete will help guide future expansions of the metadata framework. There are many example use cases one could imagine for enhanced discovery. For example, a user may wish to identify model programs that have the ability to execute using a hot start file, which may be required for a specific application like flood forecast modeling. In the current system, users can specify such details in the resource abstract as free text and/or as keywords. This reduces the metadata complexity, but if certain queries like this become a common occurrence within the system, then a new metadata element (or elements) might be needed to describe this property more precisely. Doing so, users would have the capability to more easily search and discover these resources without having to

rely on free text searches of the generic metadata fields (e.g., dc:description/dcterms:abstract). Therefore, as the system becomes more widely used, searches can be tracked, which will help guide future expansions of the metadata to better support common queries.

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

4.5 Conclusions

This work presents a model metadata framework to support discovery, sharing and interpretation of environmental models. Key features of the framework are (1) that the model program and model instance are separate concepts with a one-to-many relationship (a single model program may be linked to many model instances), (2) that metadata for these concepts extend the well-recognized and commonly used Dublin core metadata, and (3) that the model instance concept is a hierarchy with a generic parent class implementable for any model program, and a more specific level tailored for certain model programs.

A key challenge in this or any other metadata framework design is providing the right balance between rich metadata for adequately describing details of resources and minimal metadata that is critical and can be easily populated. The growing number of generic data repositories available to environmental modelers (e.g., figshare.com, zenodo.org, institutional repositories, etc.) largely adopt a minimal metadata approach. These systems provide metadata roughly equivalent to the metadata used to describe a generic resource in the HydroShare system. While this generic metadata could be used to describe, share, and discover model programs and model instances, it misses many other properties of these resources that could be leveraged for improved search, discovery, and use of model resources. Although these properties are generally included in the configuration files of the model, each model has unique configurations files, making it difficult, if not impossible, for interested users and/or an automated system to extract the pertinent metadata across models. The purpose of the metadata analysis and design presented here is to provide a more thorough, detailed metadata approach for model programs and instances. We expect to improve this metadata design over time as lessons are learned from its use, and as progress is made within the broader metadata and scientific modeling communities.

With the growing number of systems that serve a role within the larger cyberinfrastructure being built to support science, interoperability between these systems is becoming a more pressing need. If these systems are built from an agreed upon metadata framework, then it simplifies the transfer of resources between the systems. This would encourage each system to specialize in selected use cases while relying on external systems to handle other use cases outside of its scope. For example, in this work HydroShare specializes in model metadata, resource sharing, and resource publication. In ongoing research, we are building interoperability with the external SWATShare system that focuses on SWAT model execution and visualization (Rajib et al., 2016).

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

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

4.6 References

Argent, R.M., 2004. An overview of model integration for environmental applications - components, frameworks and semantics. Environ. Model. Softw. 19, 219e234. http://dx.doi.org/10.1016/S1364-8152(03)00150-6.

Billah, M.M., Goodall, J.L., Narayan, U., Essawy, B.T., Lakshmi, V., Rajasekar, A., Moore, R.W., 2016. Using a data grid to automate data preparation pipelines required for regional-scale hydrologic modeling. Environ. Model. Softw. 78, 31e39. http://dx.doi.org/10.1016/j.envsoft.2015.12.010.

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 Space Sci. 3, 226e244. http://dx.doi.org/10.1002/2015EA000142.

Elag, M., Goodall, J.L., 2013. An ontology for component-based models of water resource systems. Water Resoures Res. 49, 5077e5091. http://dx.doi.org/ 10.1002/wrcr.20401.

Essawy, B.T., Goodall, J.L., Xu, H., Rajasekar, A., Myers, J.D., Kugler, T.A., Billah, M.M., Whitton, M.C., Moore, R.W., 2016. Server-side workflow execution using data grid technology for reproducible analyses of data-intensive hydrologic systems. Earth Space Sci. 3, 163e175. http://dx.doi.org/10.1002/2015EA000139.

Gil, Y., David, C.H., Demir, I., Essawy, B.T., Fulweiler, R.W., Goodall, J.L., Karlstrom, L., Lee, H., Mills, H.J., Oh, J.-H., Pierce, S.A., Pope, A., Tzeng, M.W., Villamizar, S.R., Yu, X., 2016. Towards the geoscience paper of the future: best practices for documenting and sharing research from data to software to provenance. Earth Space Sci. 3 (10), 388e415. http://dx.doi.org/10.1002/2015EA000136.

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: open modelling interface. J. Hydroinformatics 9, 175. http://dx.doi.org/10.2166/hydro.2007.023.

Harpham, Q., Danovaro, E., 2015. Towards standard metadata to support models and interfaces in a hydro-meteorological model chain. J. Hydroinformatics 17.2, 260e274. http://dx.doi.org/10.2166/hydro.2014.061. IWA Publishing.

Heard, J., Tarboton, D., Idaszak, R., Horsburgh, J., Ames, D., Bedig, A., Castronova, A., Couch, A., 2014. An Architectural Overview of HydroShare, a Next-generation Hydrologic Information System. International Conference on Hydroinformatics. CUNY Academic Works. http://academicworks.cuny.edu/cc_conf_ hic/311.

Hill, L., Crosier, S., Smith, T., Goodchild, M., 2001. A content standard for computational models. D-Lib Mag. 7 (6), 1082e9873.

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, 4. http://dx.doi.org/10.1111/1752-1688.12363.

ISO, 2003. ISO 19115:2003 Geographic Information e Metadata accessed August 2016. http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020.

ISO, 2011. ISO 19156:2011 Geographic Information -- Observations and Measurements accessed August 2016. http://www.iso.org/iso/catalogue_detail.htm? Csnumber=32574.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., 2013. Integrated environmental modeling: a vision and roadmap for the future. Environ. Model. Softw. 39, 3e23. http://dx.doi.org/10.1016/j.envsoft.2012.09.006.

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.

Morsy, M., 2015. Rocky Branch Watershed Simulation, HydroShare. http://www.hydroshare.org/resource/12d195906f2c41918cb24e11a5c3ab60.

Morsy, M.M., Goodall, J.L., Shatnawi, F.M., Meadows, M.E., 2016. Distributed stormwater controls for flood mitigation within highly urbanized watersheds: case study for the Rocky Branch watershed in Columbia, sc USA. J. Hydrologic Eng. 21 (11), 05016025-1e05016025-10. http://dx.doi.org/10.1061/(ASCE)HE.1943-5584.0001430.

Peckham, S.D., 2014. The CSDMS standard names: cross-domain naming conventions for describing process models, data sets and their associated variables. In: Ames, D.P., Quinn, N.W.T., Rizzoli, A.E. (Eds.), Proceedings of the 7th International Congress on Environmental Modelling and Software. International Environmental Modelling and Software Society (iEMSs), San Diego, California. ISBN: 978-88-9035-744-2. https://csdms.colorado.edu/mediawiki/images/Peckham_2014_iEMSs.pdf.

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. Geosciences 53, 3e12. http://dx.doi.org/10.1016/j.cageo.2012.04.002.

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. http://dx.doi.org/10.2200/S00233ED1V01Y200912ICR012.

Rajib, Md Adnan, Merwade, V., Luk Kim, I., Zhao, L., Song, C.X., Zhe, S., 2016. A web platform for collaborative research and education through online sharing, simulation and visualization of SWAT models. Environ. Model. Softw. 75, 498e512. http://dx.doi.org/10.1016/j.envsoft.2015.10.032.

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

Sadler, J.M., Ames, D.P., Livingston, S.J., 2015. Extending HydroShare to enable hydrologic time series data as social media. J. Hydroinformatics jh2015331 18 (2), 198e209. http://dx.doi.org/10.2166/hydro.2015.331.

Singh, V.P., Frevert, D.K., Rieker, J.D., Leverson, V., Meyer, S., Meyer, S., 2006. Hydrologic modeling inventory: cooperative research effort. J. Irrigation Drainage Eng. 132, 98e103. http://dx.doi.org/10.1061/(ASCE)0733-9437(2006)132:2(98).

Singh, V.P., Woolhiser, D.A., 2002. Mathematical modeling of watershed hydrology. J. Hydrologic Eng. 7, 270e292. http://dx.doi.org/10.1061/(ASCE)1084-0699(2002)7:4(270).

Tarboton, D.G., Idaszak, R., Horsburgh, J., Heard, J., Ames, D., Goodall, J., Band, L., Merwade, V., 2014a. A Resource Centric Approach for Advancing Collaboration through Hydrologic Data and Model Sharing. International Conference on Hydroinformatics. CUNY Academic Works. http://academicworks.cuny.edu/cc_conf_hic/314.

Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodball, J.L., Merwade, V., Couch, A., Arrigo, J., Hooper, R., Valentine, D., Maidment, D.R., 2014b. HydroShare: advancing collaboration through hydrologic data and model sharing. In: Ames, D.P., Quinn, N.W.T., Rizzoli, A.E. (Eds.), International Environmental Modelling and Software Society (IEMSs) 7th International Congress on Environmental Modelling and Software doi:978-88-9035-744-2.

Taylor, P., Cox, S., Walker, G., Valentine, D., Sheahan, P., 2014. WaterML2. 0: development of an open standard for hydrological time-series data exchange. J. Hydroinformatics 16 (2), 425e446.

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

Wosniok, C., Lehfeldt, R., 2013. A Metadata-driven Management System for Numerical Modeling. In: Proceedings of OCEANS '13 MTS/IEEE. San Diego, CA, September 23e26.

Chapter 5: Conclusions and Future Work

This research addressed three challenges related to flooding impacts: (i) estimating the potential of distributed stormwater infrastructure, namely rain gardens, to mitigate flooding in urban catchments, (ii) designing and building a cloud-based real-time flood warning systems for emergency management purposes, and (iii) designing and building cyberinfrastructure to support reuse and transparency in both flood modeling and hydrologic and environmental modeling more broadly.

Chapter 2 focused on improving the understanding of how the adoption of LID practices, in particular rain gardens, at the parcel level in an already urbanized watershed might impact runoff detention and, therefore, flood risk. In this study, the storage volume added by the rain gardens was the product of two model variables: the total area of the rain gardens in the watershed as a percentage of the total impervious surface and the ponding depth (or berm height) of the rain gardens. The results suggest that implementing rain gardens with an area of 20% of the study area impervious surfaces is sufficient to mitigate flooding for storm events with less than or equal to a 10 year return period, if the maximum recommended ponding depth of 30 cm is used. It also was determined that 15% of runoff from impervious surfaces would need to be diverted to mitigate flooding for a 2 year return period, 1 hour duration storm. For a 5 year return period, 1 hour duration storm, there would need to be a 27% runoff reduction. Storms with a 10 year return period would require 38% runoff reduction, whereas higher return periods would require over 50% runoff reduction. The results of this study suggest that distributed LID approaches could potentially be used to mitigate up to a 5 or even 10 year return period storm. However, further research on possible adoption rates within the study watershed is needed to verify this conclusion.

Chapter 3 provides a design for creating a cloud-based flood forecasting system to assist transportation decision makers in time-sensitive, emergency situations. The flood forecasting system was implemented to provide decision makers with forecasts of flooded roadways and bridges based on rainfall forecasts. The 2D hydrodynamic model used in this study, which was executed for a modeling duration of 15 days, was executed up to 80x faster by using GPU resources compared to using a single CPU (from 120 hours to 1.5 hours). An automated cloud-based workflow using Amazon Web Services (AWS) resources was designed and created to link and enhance the three core model components: (i) retrieval and formatting of high resolution gridded rainfall forecast data, (ii) execution of the 2D model in a short duration to identify flood prone bridges, and (iii) real-time dissemination of model output via generation of an online map with flood warnings and the ability to send automated alert messages via email. This cloud-based approach provides an innovative way to perform flood modeling by automatically utilizing computational resources only when the flood events are likely to occur. Additionally, the workflow is automated, start to finish, without any intermediate human interaction. This work presented a preliminary calibration of the 2D model, but additional work is needed to further calibrate and evaluate the model across multiple historical flooding events. Calibration was challenging due to the scarcity of operational river gauges and significant model run-time. Results of this study suggest a higher resolution grid will improve model accuracy, but this too comes with an increased model run-time. A final challenge that needs further investigation is the differences between the 2D model outputs using CPU and GPU solvers. More research is needed to see if improving model input data, such as using a finer DEM resolution or NEXRAD rainfall data, will improve the accuracy of the GPU-based model results. Including also surveyed creek bathymetry data along with bridge structure information might also improve the model results.

Chapter 4 presents a model metadata framework to support discovery, sharing and interpretation of environmental models. Key features of the framework are (i) the model program and model instance, which are separate concepts, are linked in a one-to-many relationship (a single model program may be linked to many model instances), (ii) metadata for these concepts, which extend the well-recognized and commonly used Dublin Core metadata framework, and (iii) the model instance concept, which is a hierarchy with a generic parent class implementable for any model program, and more specific child classes tailored for certain model programs. A key challenge in this or any metadata framework design is providing the right balance between rich metadata for adequately describing details of resources and minimal metadata that is critical and can be easily populated. While generic metadata could be used to describe, share, and discover model programs and model instances, it misses many other properties of these resources that could be leveraged for improved search, discovery, and reuse of model resources. The purpose of the metadata analysis and design presented in this work is to provide a more thorough, detailed metadata approach for model programs and instances. With the growing number of systems that serve a role within the larger cyberinfrastructure being built to support science, interoperability between these systems is becoming a more pressing need. If these systems are built from an agreed upon metadata framework, then it simplifies the transfer of resources between the systems. This would encourage each system to specialize in selected use cases while relying on external systems to handle other use cases outside of its scope. Future work will be aimed at improving the usability of the model program and model instance resources within HydroShare. For example, to reduce the time spent manually completing metadata fields, new functionality is planned to automate metadata extraction when a resource is uploaded and the metadata are already present within files uploaded with the resource. Automatic metadata extraction, along with the increased use of

controlled vocabularies, would increase the usability of the system for both sharing and discovery. This approach is difficult, however, given the diversity among environmental models; extracting metadata directly from model input files may require a significant amount of custom code.

In conclusion, this dissertation presents new tools and approaches to assist decision makers in better understanding, addressing, and finding solutions for flooding problems faced on the catchment and regional scales. It also demonstrates the potential benefit of harnessing the rapidly advancing cloud and cyberinfrastructure technologies to advance hydrologic modeling. Through computational models, this work (i) contributes understanding of the potential of rain gardens as distributed stormwater control for flood mitigation at the catchment-scale and (ii) it demonstrates the use of the cloud for building an automated regional-scale flood warning system using a sophisticated 2D hydrodynamic model. This work also contributes a solution for sharing and reusing hydrologic models, allowing them to be documented and shared with other scientists, as well as with decision makers, which encourages model reusability, reproducibility, and transparency.