A Software Environment for Watershed Modelling

Moinul Hossain, Sergiu Dascalu and Frederick C Harris, Jr.
Dept of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, USA
hossain@nevada.unr.edu, dascalus@cse.unr.edu, fredh@cse.unr.edu

Abstract

Watershed scientists use different computer based modelling tools to model the physical processes around watersheds. In this work we have created a software environment for assisting watershed scientists in their research. This software environment will provide a platform to share different data with fellow scientists and run different models in the cloud through web services. It will also provide option for the scientists to have social discussion on the data and the models created and ran in the system.

keywords: software environment; watershed; web services; modelling

1 Introduction

Modelling of physical processes is a core part of the scientific enquiries. Scientists in all domains including earth science build computer models to investigate physical phenomena. Softwares are becoming a very important part of the modern scientific research as a result. Quality, scalability and maintainability are big concerns for scientific softwares, but there has not been a substantial amount of research in the community on creating sustainable softwares by adhering software engineering principles. Software processes like agile practice or test driven development are not much popular in the area of academic scientific softwares as they are in the area of commercial softwares. As a result softwares in academia end up being ad-hoc products that are not compatible with other research products. This is a problem in the area of scientific modeling as well. Model coupling is a hard problem due to the the different reasons including nature of data or the process being modelled. Issues like data storage, retrieval, running and coupling models are hard problems and require extra care from the perspective of software engineering. Designing integrated systems that provides means to handle all these issues can be a challenging job.

Building software tools and frameworks for scientific research can be interesting for many reasons. With the advancement of computing power in recent decades, scientific research is creating more and more data and models independently built by scientific researchers. This is an exciting field where software engineering can assist this emergence by facilitating the creation of distributed software systems and frameworks to assist scientists have collaboration on these data and models in a global scale. Another great aspect of this field is, being an interdisciplinary field it poses lots of challenges in terms of barriers of communication and team building among different communities involved in the process.

The rest of the sections include related works done before in the area of software system for assisting scientific modelling, the problem description, the details description of the design and finally some initial results and prototypes.

2 Adhering to Software Engineering Principles in Building Scientific Software

The most challenging job in conducting software engineering research in collaboration with scientific community is how to adopt the software engineering processes and principles. Building software systems for scientific community is different from building commercial softwares in many different ways, thus it poses numerous obstacles making softwares following the software engineering way. There has been numerous studies on how software engineering principles can be adapted in a collaboration project with scientists.

Scientific research in academia often under appreciated the use of software methodologies in developing research tools [1]. Researchers spend enormous amounts of time developing niche, ad-hoc software solutions to answer their individual research questions which are not applicable to other research questions, datasets, or

978-1-943436-01-9 / copyright ISCA, SEDE 2015
October 12-14, 2015, San Diego, California, USA

researcher needs. Numerous studies on developing the quality of software products made for scientific research by adapting software engineering processes has been conducted by several research groups.

Sletholt et al. discussed on adoption of agile practices in scientific software development [7]. Software Engineering best practices and research have rarely been adopted in software developed by scientists for scientists. The authors surveyed and analyzed how much and how well current scientists employ best software engineering practices for scientific computing. In particular the authors investigated the use of agile practices for scientific software development. The authors investigated several use cases of scientific software development projects that adopted some form of agile practice. A comparative analysis across multiple use cases is presented. Common agile practices are compared across these selected projects to show what practices are adopted in which projects. The overall finding of this study shows that agile practices are not adhered by the project teams with a few exception of some self organized teams.

Another study by Christopherson et al. talks about developing quality software for scientific research through community engagement process [2]. The quality of academic software, tends to be lower than commercially-developed software because of the lack of adoption of software process like test driven development. The authors investigated different barriers in engaging software process in scientific software development process and proposed their method of community engagement process.

Wilson describes a different approach called software carpentry rather than software engineering to help scientific researchers write their codes in a more engineered way by following the software engineering principles [10]. The author created a 150 hours long course to help the scientists understand the need of following software process in their development process. The authors conclude that the barriers in adhering software engineering principles in scientific software development are more social than technical.

Another work by Segal describes a case study of software engineers developing a library of software components for a group of research scientists, using a traditional, staged, document-led methodology [6]. The case study reveals two concerns with the use of the methodology. The first is that it demands an upfront articulation of requirements, whereas the scientists had experience, and hence expectations, of emergent requirements; the second is that the project documentation does not suffice to construct a shared understanding.

3 Related Software Products

There has been numerous research on creating software frameworks and environments to facilitate scientific research by different interdisciplinary research groups. Several successful collaborative research work on software frameworks and environments in the fields related to earth science are discussed in brief in this section.

3.1 CSDMS: The Community Surface Dynamics Modeling System

The Community Surface Dynamics Modeling System (CSDMS) project started in 1999 to facilitate earth surface modellers by creating a community driven software platform. CSDMS applies a component-based software engineering approach to the integration of plug-and-play components as development of complex scientific modeling system requires the coupling of multiple, independently developed models [5]. There are several benefits CSDMS brings to the community of modellers. Firstly, it provides means for the modellers from different backgrounds to write their components in any of the popular languages. CSDMS achieves this via language interoperability using Babel language interoperability tool. CSDMS treats components as precompiled units which can be replaced, added to, or deleted from an application at runtime via dynamic linking. This gives modellers the option to use components created by others in the community to use in their simulation easily. The key design criteria that drove the design of CSDMS include support for multiple operating systems, language interoperability across both procedural and object oriented programming languages, platform independent graphical user interfaces, use of established software standards, interoperability with other coupling frameworks and use of HPC tools integrate parallel tools and models into the ecosystem.

3.2 HydroShare

The Consortium of Universities for the Advancement of Hydrologic Science, Inc (CUAHSI) is one of the leading research organizations representing universities and international water science-related organizations to develop software infrastructure and services for advancing water science. CUAHSI has several software projects such as HydroShare and CUAHSI HIS to provide infrastructure for water science research.

HydroShare is an online, collaborative software system for sharing hydrologic data and models. The goal of HydroShare is to help scientists to discover and access data and models, retrieve them to their

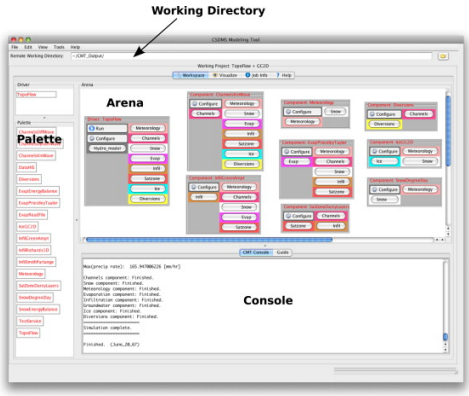

Figure 1: The graphical user interface of CSDMS Modelling Tool (CMT) where modellers can setup and run their models to experiment with minimal effort [5].

desktop or perform analyses in a distributed computing environment that may include grid, cloud or high performance computing model instances [8]. Scientists can also publish outcomes of their research whether its data or model into HydroShare, using the system as a collaboration platform for sharing data, models and analyses with other modellers. HydroShare is built using a python based technologies. The main components are built on top of Django Web Application Framework. Django is an application framework to facilitate web oriented architectures. It has several components designed for the end users to communicate with the system. HydroDesktop, a web service based desktop client is designed for the end users for hydrologic data discovery, download, visualization, and analysis.

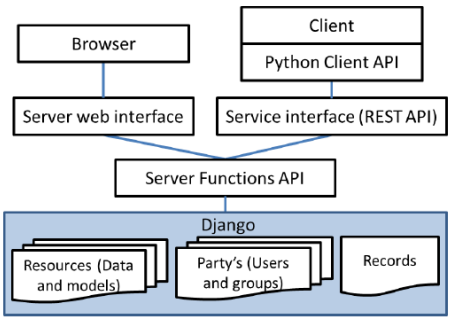

Figure 2: The high level architecture of HydroShare [8].

The architecture of HydroShare separates the web application interface layer from the service layer, exposing the functionality through an application programming interface (API) to enable direct client access and interoperability with other systems [8].

3.3 Other Related Products

McGuire and Roberge designed a social network for assisting collaborations between watershed scientists [4]. Being highly available, hydrologic data has not been integrated in a single system and no system exists to facilitate collaboration for scientists, citizen scientists, and the general public. This work presents the design of a collaborative social network aimed at multiple user groups who are focused on hydrology and watershed science.

The Demeter Framework by Fritzinger et al. is another attempt to bring software framework for assisting scientists in the area of climate change research. This work presents an overview of a software framework named the Demeter Framework that proposes a new solution to the model coupling problem by taking a component-based approach that allows almost any standard or type of component to be integrated within the system.

Walker and Chapra proposed a web based client-server approach to solve the problem of environmental modeling compared to the traditional desktop based approach [9]. With improvement in modern day web browsers, client-side approaches allow for improved user interfaces that is better than traditional desktop softwares, as well as the ability to perform simulations and visualizations within the browser.

The Geographic Storage, Transformation and Retrieval Engine (GSToRE) is a project initiated by the Earth Data Analysis center at the University of New Mexico which provides a data framework for data discovery, delivery and documentation for scientific research specializing in earth science. It has been developed as a flexible, scalable data management, discovery and delivery platform that supports a combination of open and community standards. It is built upon the principle of a services oriented architecture that provides a layer of abstraction between data and metadata management technologies.

4 The Problem

The main goal of this work is to provide a software environment for the watershed scientists to run their models, share their data and models and have social interaction with the fellow scientists. The key functionalities include:

- Ability to share data and model with other scientists with multiple level of privacy
- Ability to run the models in cloud through a web client

- Ability to have social interactions to on the data and the models with fellow scientists.

The key concern is the software should be an easy to use and accessible for the scientists to share their knowledge. There are numerous challenges needs to be solved in order to build such software environment. Different scientists use different models not only in terms of data formats but also programming language, OS platform etc. Designing a generic interface to share the data is an important task to tackle. Designing a generic interface for running the models requires substantial amount of research as well.

5 The Design of the System

Cloud and micro service based solutions are becoming more and more popular. We have proposed cloud based solution to the problem of data storage and model run integration. Figure 3 shows a depiction of the architecture. The main idea behind the design is to expose different models as web services.

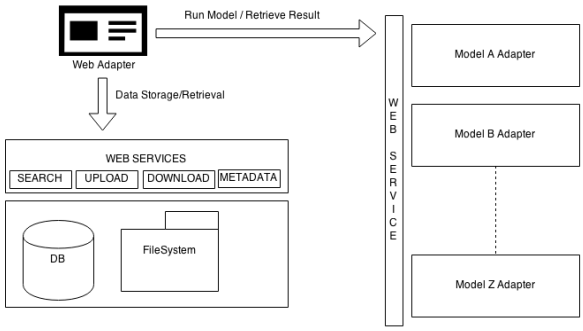

Figure 3: The main architecture of the system.

Clearly we have two different components in our system. One is storage component for the data and another is computation components for the models. A web service wrapped around databases, file system and a search engine can act as the end point for data storage and retrieval. The main advantage of this approach is we can expose data in different formats allowing to use data converters intercepting the web service to provide data as needed for different models.

The other components in system are the models which are computing resources. As models have differences in terms of programming language, file format, os etc the best way to expose models are through web services as well. Exposing 'models as a service' gives the benefit of having platform independent cloud ready system. For that purpose the models needs to be wrapped around with a service adapter. Each model can live in its own computing node exposing the model capabilities though a common web service interface. A thin client (e.g., a web application) can be easily created to extract data and run some model on the data by using the common web service interface. Figure 4 shows the architecture of a model server. Each model server contains self contained web service wrapper with its own lightweight database to keep track of model runs and a model adapter is wrapped around the model to implement the common web service interface. Figure 4 shows couple of possible web service methods as well.

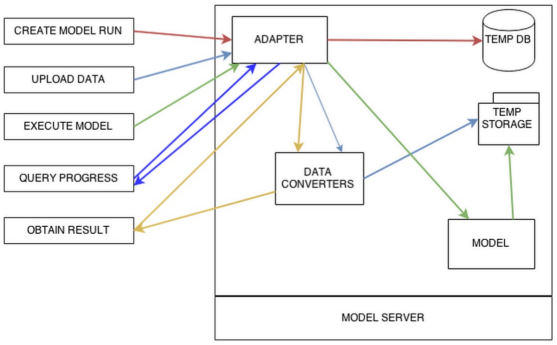

Figure 4: The design of a model server

6 Results and Discussions

We have created a web application client to demonstrate the idea as a proof of concept of the design. The web application works as a thin client of the system where it utilizes different web services to accomplish tasks like data storage and retrieval and model runs. Figure 5 shows a simple interface for searching data in the system.

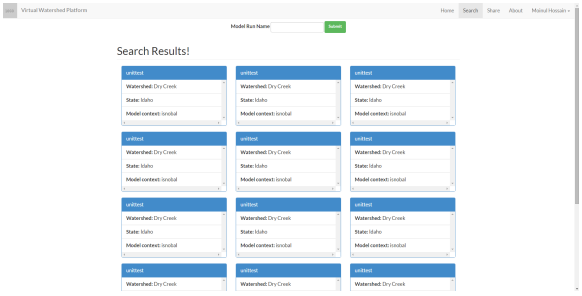

Figure 5: The search interface of the system

A search client communicates with the data server through web service to retrieve the metadata of the data shared by the other users. User can wish to download the data in a specified format by invoking a data converter tool. The system provides a generic web service interface so that users can register their own

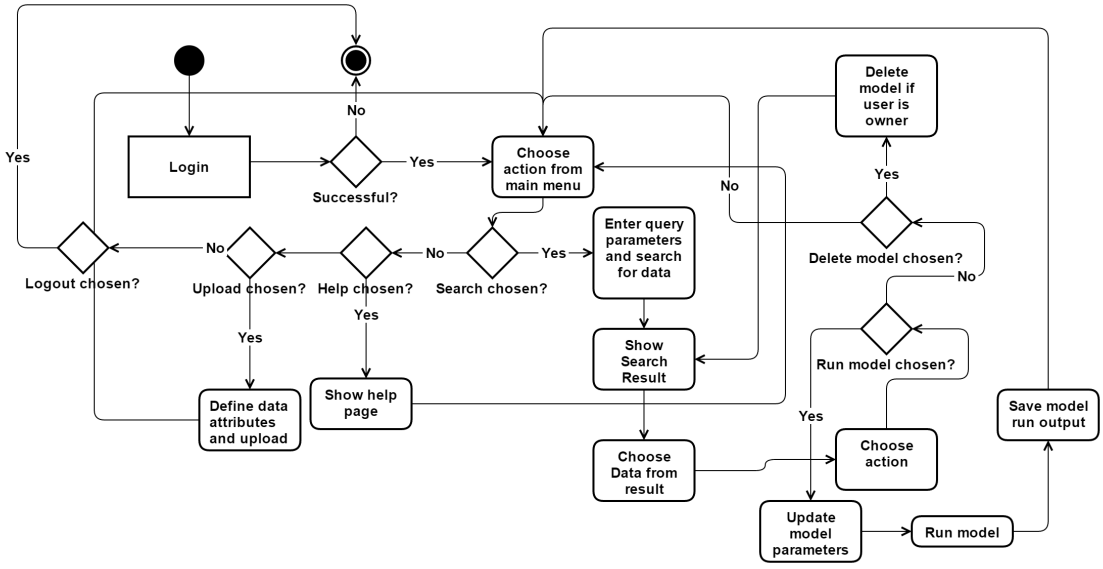

Figure 6: The activity flow of the system

data converter that they wish to use. Figure 7 shows a data upload / download interface of the system.

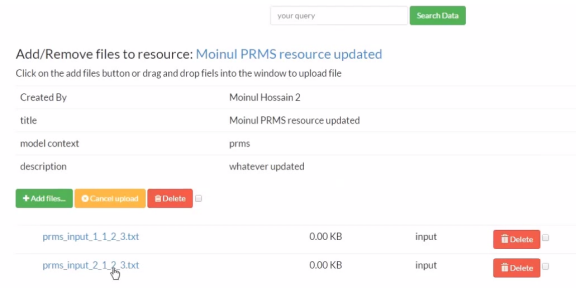

Figure 7: The data upload / download interface of the system

A user can wish to upload /download data of a model by simply drag and drop the data files into browser. The storage server is communicated through the web service to transfer / receive the data.

Figure 8 shows the interface for model run. A user can invoke a model on a dataset through a single click of the mouse setting up the parameters for a run. This is another benefit of the architecture where the users are not required to setup their local instance of the models in order to run them with some data. If a model is once registered with the system through web services it can be used to run on datasets without have to worry about setting up.

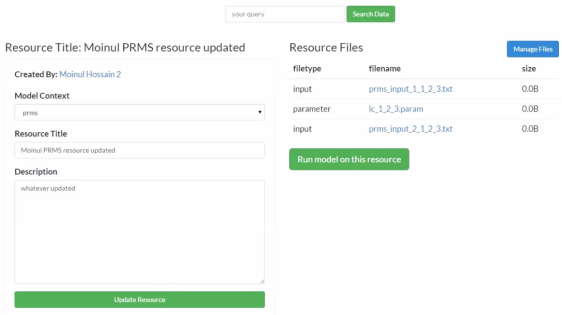

Figure 8: The model run interface of the system

7 Future Work and Conclusion

We have tried to create a generic extensible environment for watershed science research. The main contribution of this work is the design of web service based components which makes the system scalable and easily integrable between components. The system is currently in a prototype stage and requires more care to make it more robust. Currently the architecture is monolithic for model run through web services. Future works include making the model run interface distributed as micro services to make it more robus and accessible. And allowing users to register their own data converters through web services will also make the system more usable in terms of data storage and

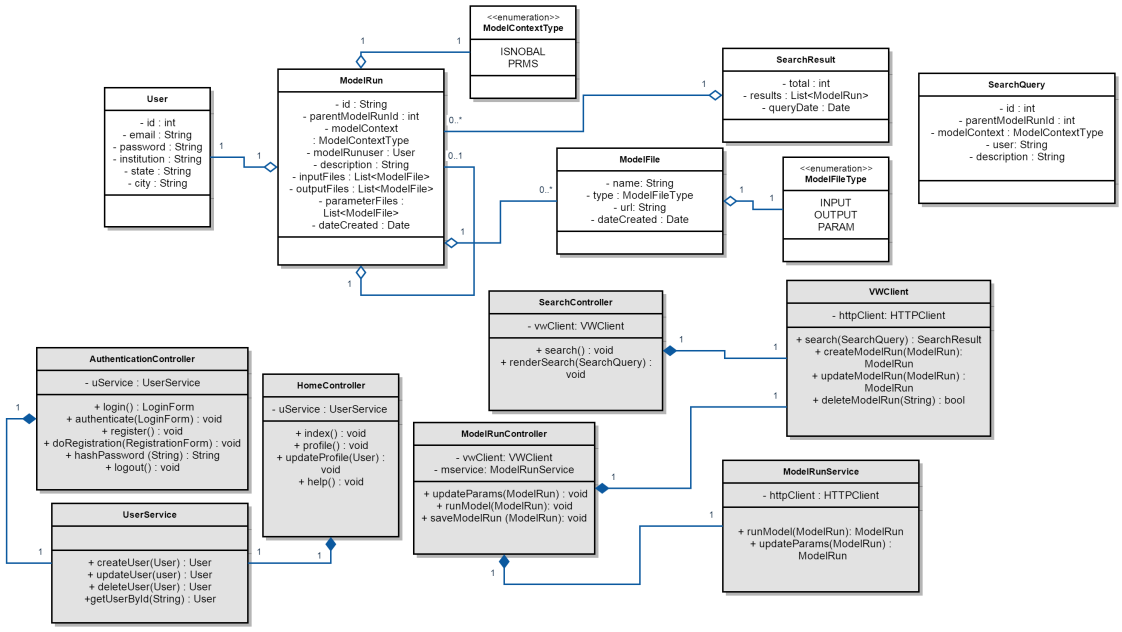

Figure 9: The class diagram of the system

retrieval.

References

[1] Stan Ahalt, Larry Band, Laura Christopherson, Ray Idaszak, Chris Lenhardt, Barbara Minsker, Margaret Palmer, Mary Shelley, Michael Tiemann, and Ann Zimmerman. Water science software institute: Agile and open source scientific software development. *Computing in Science & Engineering*, 16(3):18–26, 2014.

[2] Laura Christopherson, Ray Idaszak, and Stan Ahalt. Developing scientific software through the open community engagement process. Technical report, Technical Report 790723, figshare, 2013. http://dx. doi. org/10.6084/m9. figshare. 790723.

[3] E Fritzinger, S Dascalu, DP Ames, K Benedict, I Gibbs, Michael J McMahon Jr, and Harris FC Jr. The Demeter framework for model and data interoperability. In *Proceedings of the International Congress on Environmental Modeling and Software (IEMSS-2012)*, pages 1535–1543, 2012.

[4] Michael P McGuire and Martin C Roberge. The design of a collaborative social network for watershed science. In *Geo-Informatics in Resource Management and Sustainable Ecosystem*, pages 95–106. Springer, 2015.

[5] Scott D Peckham, Eric WH Hutton, and Boyana Norris. A component-based approach to integrated modeling in the geosciences: The design of CSDMS. *Computers & Geosciences*, 53:3–12, 2013.

[6] Judith Segal. When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4):517–536, 2005.

[7] Magnus Thorstein Sletholt, Jo Erskine Hannay, Dietmar Pfahl, and Hans Petter Langtangen. What do we know about scientific software development's agile practices? *Computing in Science & Engineering*, 14(2):24–37, 2012.

[8] DG Tarboton, R Idaszak, JS Horsburgh, D Ames, JL Goodall, LE Band, V Merwade, A Couch, J Arrigo, RP Hooper, et al. Hydroshare: an online, collaborative environment for the sharing of hydrologic data and models. In *AGU Fall Meeting Abstracts*, volume 1, page 1510, 2013.

[9] Jeffrey D Walker and Steven C Chapra. A client-side web application for interactive environmental simulation modeling. *Environmental Modelling & Software*, 55:49–60, 2014.

[10] Greg Wilson. Software carpentry. *Computing in Science & Engineering*, 8:66, 2006.