Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 1731–1740

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

A Web-based Visual Analytic Framework for Understanding Large-scale Environmental Models: A Use Case for The Community Land Model

Yang Xu[1,2], Dali Wang[3]*, Tomislav Janjusic[4], Wei Wu[5], Yu Pei[5], and Zhuo Yao[5]

[1] Department of Geography, University of Tennessee,Knoxville, TN, USA
[2] Senseable City Laboratory, SMART Centre, Sinagpore
yangxu@smart.mit.edu
[3] Climate Change Science Institute,
Oak Ridge National Laboratory, Oak Ridge, TN, USA
wangd@ornl.gov
[4] Computer Science and Mathematics Division,
Oak Ridge National Laboratory, Oak Ridge, TN, USA
janjusict@ornl.gov
[5] Department of Electrical Engineering and Computer Science,
University of Tennessee, Knoxville, TN, USA
{wwu12;peiy;zyao}@vols.utk.edu

Abstract

This study introduces a web-based visual analytic framework to better understand the software structures of large-scale environmental models. The framework integrates data management, software structures analysis, and web-based visualizations. A system for the Community Land Model (CLM) is developed to demonstrate the capability of the proposed framework. It consists of three major components: (1) a Fortran-syntax analysis tool that decomposes CLM source code into simpler forms; (2) an application tier that further analyzes and converts the preprocessed data into meaningful software structural information; (3) a web-based front end that is developed using state-of-the-art web technologies and visualization toolkit (e.g., D3.js). The framework provides users with easy access to the internal structures of complex environmental models. Currently, the prototype system is being used by CLM modelers and field scientists to tackle different environmental research problems.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: software engineering, environment, web-based services, information visualization

*Corresponding author. POB 2008, MS 6301, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA. Email: wangd@ornl.gov, Tel: 18654218679

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.181

1 Introduction

Over the past several decades, many computer models have been developed to gain insights into environmental systems and to explore better options for system-wide management. With the rapid development of computing technologies, many high performance, integrated environmental modeling systems have been developed to address novel research challenges. These large scale and integrated models have advanced our understanding of environmental systems. However, as the scale of these systems increases, software complexities quickly become a barrier for model interpretation and further improvements [5, 6]. Such complexities are partially reflected by the enormous number of functions and variables, and their connectivity across different modules/subroutines. A limited understanding of their relationships could hinder integrated and collaborative model developments. Moreover, the intricate relations among various model components make it difficult for empiricists (e.g., field scientist) to evaluate the inner workings of these models, which leads to a disconnection between the fundamental environmental processes and the actual model representations [8, 1]. How to provide generic and effective means that could help these users to better understand the software structures of large-scale environmental models is important to the models' longevity and applicability.

During the past two decades, many tools have been developed to provide solutions for static code analysis and/or software structure visualizations. For example, Müller et al. developed the Rigi system, which is able to identify system blocks and visualize their hierarchies in an interactive manner [2]. Later the ShriMP toolkit developed by Storey and Müller was incorporated into the Rigi system to support the visualization of system architectures at multiple levels of abstraction [4]. There are also platforms such as Understand (https://scitools.com) and Moose (http://www.moosetechnology.org) that provide generic static code analysis and visualization functions for various programming languages. Several tools have also been developed to facilitate source code understanding and documentation (such as Doxygen) and performance improvements (such as SCOREP (www.score-p.org)), however, adopting these tools to understand large scale environmental models remains challenging. Large scale environmental models (commonly in Fortran) usually have many contributors from different organizations - each with distinct knowledge and programming practices. Therefore, the software systems of these large-scale environmental models differ significantly and are largely determined by the modeling conventions among developer communities .

In this research, we introduce a web-based visual analytic framework to better understand the software structure of large-scale environmental models. The proposed framework integrates data management, software structure analysis, and advanced web-based visualizations. The purpose is to offer an integrated platform that is capable of analyzing software structures of large-scale environmental models, and providing end users with easy access to analysis results via intuitive visualizations. A prototype system for the Community Land Model (CLM) [3] is used as a case study to demonstrate the feasibility of the system design.

2 The Software System of CLM

The Community Land Model (CLM) is the land component for the Community Earth System Model (CESM). CLM simulates several aspects of the land surface, including surface heterogeneity, land biogeochemistry, biogeophysics, human dimensions, hydrologic cycle and ecosystem dynamics. CLM has a long development history, which can be tracked back into the official release in 1996 (http://www.cgd.ucar.edu/tss/lsm/) on several targeted Cray machines. The latest release of the model is CLM version 4.5, which contains several submodels includ-

ing Canopy Fluxes, Ecosystem Dynamics, Hydrology, Urban, Soil, Temperature, Fire, Dust, etc. The structure of each submodel is organized based on natural system functions such as carbon-nitrogen cycles, hydrology, photosynthesis, and soil temperature. Each submodel interacts with a list of variables which are globally accessible or subroutine explicit. The whole CLM simulation system contains more than 2100 files and over 380,000 lines of code. The software system contains both physical earth system components (e.g., land, atmosphere, ocean, ice, and glacier) and software components for coupled simulation, including an application driver, a flux coupler, as well as several shared software modules and utilities for computer IO (input/output) and parallel computing. The software complexity of CLM simulation system quickly becomes a barrier for future developments [7].

3 A Web-based Visual Analysis System for CLM

We propose a web-based visual analytic framework using a three-tier architecture to manage, analyze and visualize the software structures of CLM. Figure 1 illustrates the generic design of the system architecture. On the data tier, a database is deployed to host the information of software structures pre-processed from CLM source code. The application tier serves as a bridge between data tier and client tier. Specifically, the Python CGI module (Common Gateway Interface Support) is used to process HTTP requests from the client side. After receiving the requests, the Python scripts will perform further analysis to extract meaningful software structure information and then send the results back to the clients via JSON (JavaScript Object Notation). The client side is implemented using standard web developing tools (e.g., HTML5, JavaScript and CSS) and open source libraries including D3.js and jQuery (http://jquery.com). Three major functions (software structure overview, relational query of functions/variables, and visualization of submodel structure) are incorporated to support dynamic query and interactive visualizations of CLM software structures.

3.1 Software Decomposition and Organization of Data Tier

To decompose the CLM software structure into simpler forms, a Fortran-syntax analysis tool was developed to categorize key CLM variables and data structures into identifiable tokens [10]. A token refers to any source-code identified function call or variable, which includes names of subroutines, globally visible variables, as well as variables used in subroutine definitions (i.e., subroutine-in, subroutine-out). Subroutine-in variables refer to the tokens identified in the subroutines' (function's) signature, and subroutine-out variables correspond to a subset that was identified to be written to. Globally visible variables are identified using the pointer assignment syntax during source-code scanning. Any token in the source code that adheres to the general pointer assignment syntax is treated as globally visible variables. The global variables are further divided into three categories (Read-only, Write-only, or Modified). During the scanning process, the source code lines are decomposed into lefthand (lHand) and right-hand (rHand) statements. Every token found on the lHand side is a Write category and similarly every token on the rHand side is a Read category. If a token falls into both categories, we will identify it as Modified. Tokens that are identified by special keywords (e.g., call) are used to extract function calls, which refer to the names of the subroutines that are used by the caller subroutine.

The output of the source code analysis is organized as a nested structure on the data tier (see Figure 1). In particular, the database stores the software structure of different versions of CLM (e.g., CLM_45_10, CLM_45_68 and CLM_45_Microbe). As shown in Figure 1, each

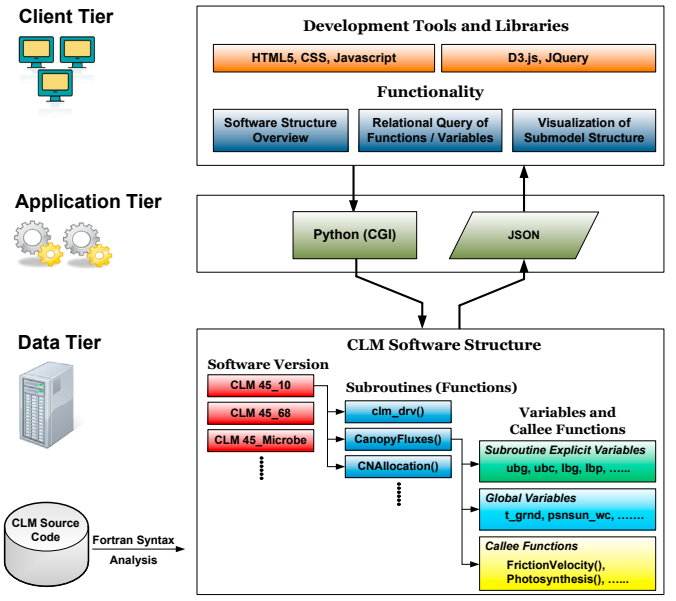

Figure 1: System architecture of the web-based visual analytic system

CLM version includes a list of tables named after the subroutines. A table associated with a subroutine stores all the variables and callee functions (i.e., child subroutines) that have been used by this subroutine. Table 1 gives an example of the tokens identified from the subroutine *CanopyFluxes()* for version CLM_45_10.

Category	Tokens
Subroutine-In	*ubg, ubc, lbg, lbp, num_nolakep, ...*
Subroutine-Out	*Null, ...*
Global Read Only	*t_grnd, psnsun_wc, alphapsnsun, psnsun, ...*
Global Write Only	*cgrnd, psnsun, rb1, ulrad, dlrad, ...*
Global Modified	*displa, rc13_psnsun, z0qv, z0hv, ...*
Functions Calls	*QSat, FrictionVelocity, Photosynthesis, ...*

Table 1: Identified tokens of *CanopyFluxes()* subroutine

3.2 Configuration of Application Tier

The application tier manages the information exchange between the data tier and the client tier. To support dynamic query and interactive visualizations of CLM software structures, we use Python as the scripting language on the server side. Specifically, the *FieldStorage* class of Python CGI module is used to handle HTTP requests from the clients. When a client sends a HTTP request to the server, the Python script will decode the query string in the submitted form, perform necessary analysis based on the information on the data tier, and then send the results back to the client as a JSON format. The JSON object can be easily processed on the client side using interfaces provided by D3.js and jQuery.js.

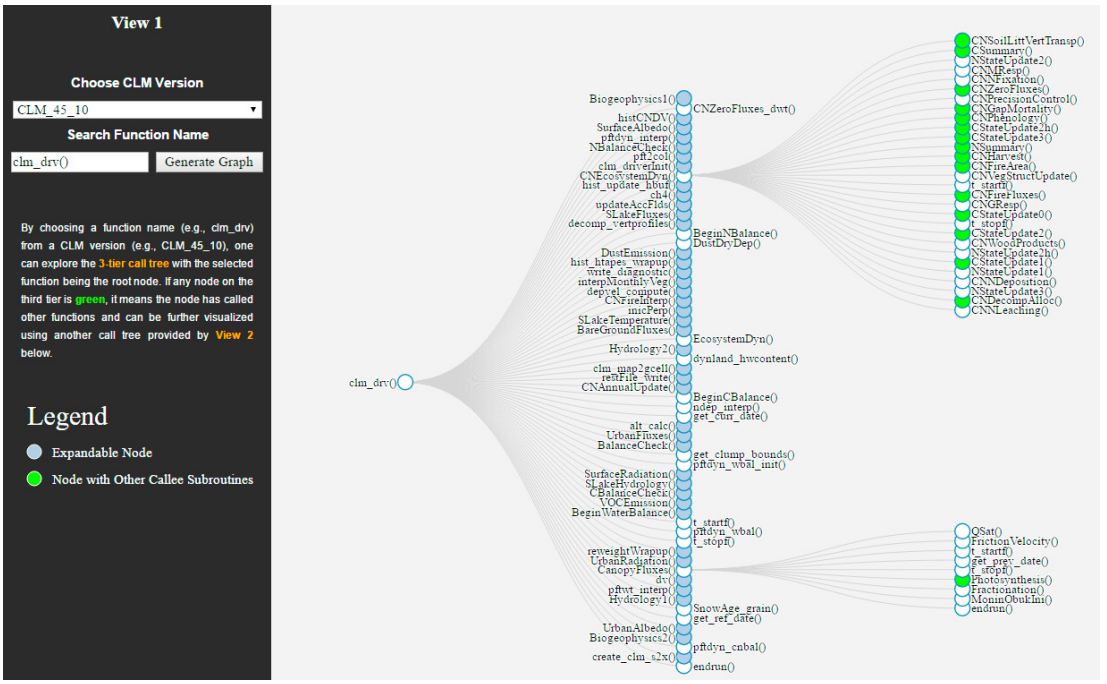

Figure 2: User interface for Software Structure Overview (only View 1 is shown to save space).

3.3 Front End Design and System Functionality

Working with large-scale environmental models requires not only adequate knowledge of software structure at the macro level, but also a good understanding of how different elements interact with each other at the micro level. The system functionality needs to be flexible enough to facilitate software structure exploration from different perspectives. To achieve this, we incorporate three major functions into our system, which are *software structure overview*, *relational query of functions/variables*, and *visualization of submodel structure*. The prototype for the system is available at `http://cem-base.ornl.gov/CLM_Web/CLM_Web.html`.

The first function of our web application is software structure overview. The function is designed to allow users to explore the call graph of CLM from an overall perspective. As illustrated in Figure 2, through the graphical user interface (GUI), one can choose any function (e.g., clm_drv) from a software version (e.g., CLM_45_10) and generate a 3-layer call tree. The selected function serves as the root node of the call tree and can be expanded to view its child node (callee functions) on the second and third layer. The nodes on the third layer cannot be further expanded in the visualization canvas but are marked with different colors to indicate if theyve called other functions in this software version (e.g., a green node indicates that it has called other functions and can be further visualized using another call tree). Our website provides two views (View 1 and View 2) with identical functionality that allows users to generate two different call trees at the same time. The dual view is a compelling feature that not only keeps the visualization compact, but also improves the flexibility of software structure exploration. For example, users can trace any node on the third tier on View 1 and further generate its call tree on View 2. They can also compare the call trees of the same function (e.g., clm_drv) between two different software versions (e.g, CLM_45_10 and CLM_45_68).

The second function of this application is designed to provide usage patterns of software functions and variables. This function includes two features (Search Downward and Search Upward) of which the functionality complements each other. By specifying a function name from a particular software version, the Search Downward feature will return a table recording all the functions and variables that have been used by the selected function. One can also use the Search Upward feature to explore how a function or variable is used by other functions in a particular software version. By combining the two features, the flexibility of the query is improved and users can locate functions/variables quickly through "top-dow" or"bottom-up" approach. (More illustration of this function can be found in Section 4).

For large-scale environmental models like CLM, a good representation of submodel structures could yield additional insights into the complex relationships among particular ecosystem processes. The third function is designed to help users better understand the interplay among different software components. By using the control widgets on top left part of the GUI, a user can choose a particular software version, search any function by name and add all the functions they want to examine into a list. Once the user clicks the *Generate Graph* button, a graph structure that summarizes the interrelationships among all related function calls and variables will be presented. The graph includes many nodes with different colors showing their categories. Also, detailed information for the node which the mouse sits on will pop up (e.g., the name and explanation for a variable/function). Users can get a clear idea of how different components are coupled together as well as the role each of them is playing. Like the first function of this web application, the way we visualize the submodel structure gives users full control of what they want to explore. For example, a user can select all the function calls related to hydrological cycle and explore how they interact with each other in the modeling context. Similarly, one can also generate a graph structure involving multiple ecosystem processes (e.g., carbon cycle and nitrogen cycle). The scale of the graph is customized by the user based on the study or application purposes. (More illustration of this function can be found in Section 4).

4 Towards A Better Understanding of CLM Software Structure

By using the key functions described in the previous sections, we can improve the understanding of CLM software structure in many convenient ways, such as tracking the usages of global variables and comparing the key model structures. In this section, we present three cases to demonstrate these capabilities.

4.1 Essential Model Structure Comparison

Many scientific communities around the world have been actively developing CLM model to address upcoming climate change issues associated with the land-atmospheric interactions. Modelers have worked to improve the representation of biogeochemistry of terrestrial ecosystem dynamics (with functional group CNEcosystemDyn). One of the examples is the consideration of nitrogen leaching. As illustrated in Figure 3, the call graphs generated by our website clearly represent the key model structure changes. The left panel shows the model structure of CNEcosystemDyn in one of the older CLM version (CLM_45_10), and the right panel shows the new functional group (CNEcosystemDynLeaching) has been added into the model to improve the Nitrogen and Carbon calculations after the computation of the original functional group, which is renamed as CNEcosystemDynNoLeaching in a later version of CLM (CLM_45_68).

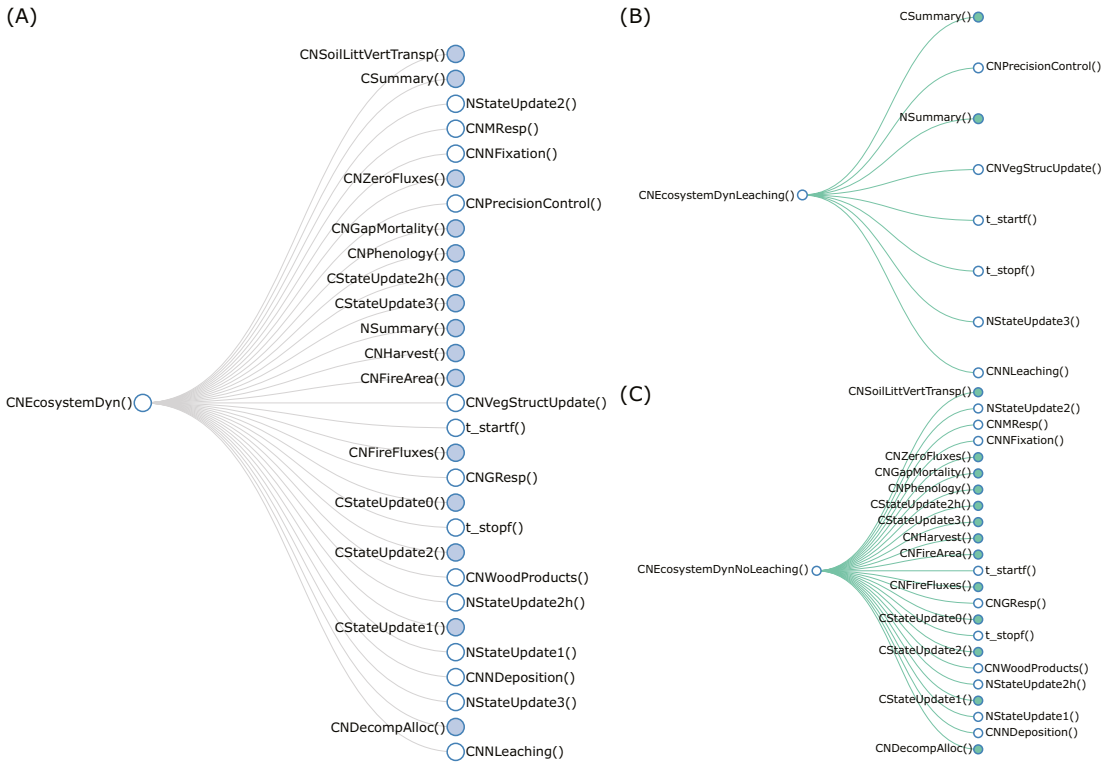

Figure 3: (A) The call graph of CNEcosystemDyn subroutine in an early version of CLM (CLM_45_10); (B) The new CNEcosystemDynLeaching functional group in a later version of CLM (CLM_45_68); (C) In this new version (CLM_45_68), the old subroutine (CNEcosystemDyn) is renamed as CNEcosystemDynNoLeaching. The visualization capability provided by the website improve modelers understanding of essential differences among model structures.

4.2 Exploration of Global Variable Usage

Our website can also facilitate the exploration of global variable usage within the CLM software system. One of CLM model development is the consideration of microbial contributions to the carbon and nitrogen decomposition and allocation. Our website can track these improvements through the global variable usages related to a key functional group related to carbon-nitrogen decomposition and allocation (i.e., CNDecompAlloc). As illustrated in Figure 4A, in an early version of CLM (CLM_45_10), the CNDecompAlloc function only contains 11 global read-only variables, 6 writeonly variables, and 6 modified variables. While in a later CLM version with microbe component (CLM_45_Microbe), CNDecompAlloc contains 14 global read-only variables, 12 write-only variables, and 11 modified variables (Figure 4B). Examples of key microbe-related global variables added include new global variables that record the microbes contributions to carbon and nitrogen pools and decompositions (e.g., cmicbions, cn_microbe, and floating_cn_ratio_decomp_pools).

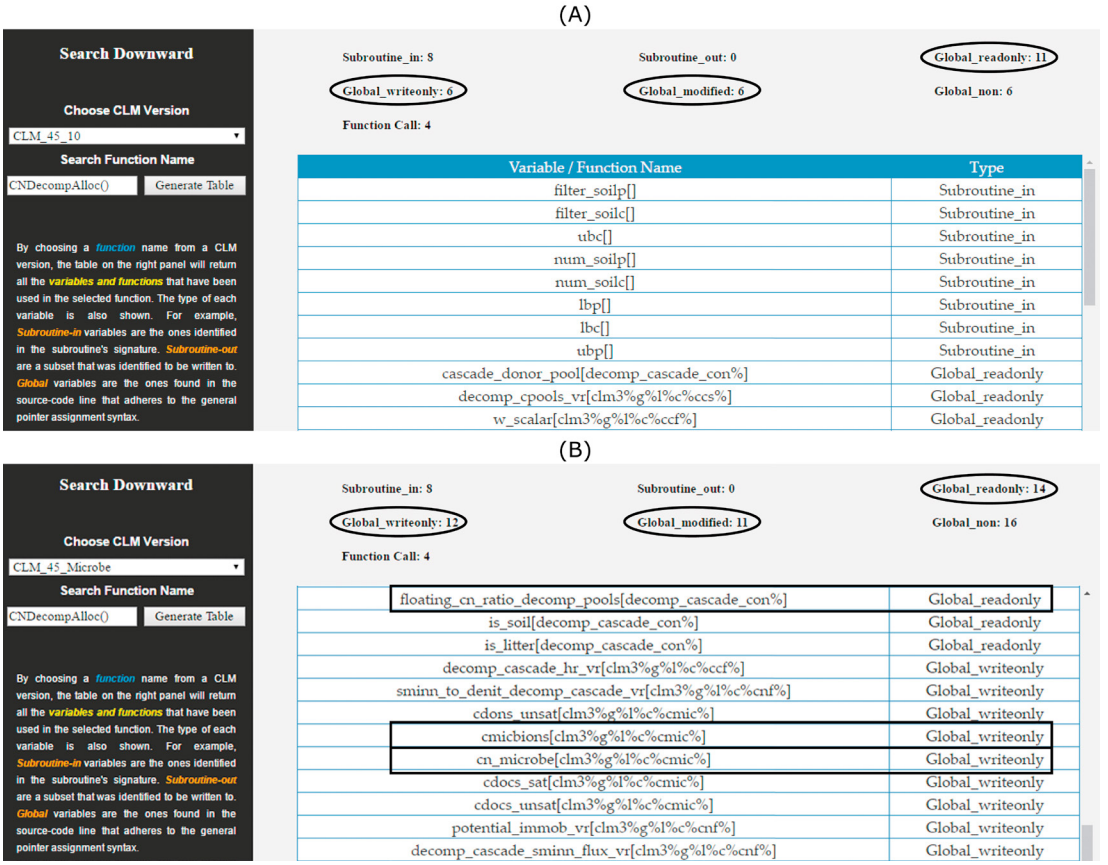

Figure 4: (A) Query result of the variables/functions that are used by CNDecompAlloc in CLM_45_10; (B) Query result of the variables/functions that are used by CNDecompAlloc in CLM_45_Microbe.

4.3 Illustrations of Submodel Structure Changes

Our software can also help modelers to identify the submodel structure changes. For example, A newer CLM version (i.e., ACME_CLM_V0) incorporates new concepts from Ecosystem Demography (ED) model as an alternative submodel, in addition to photosynthesis submodel based on the Plant Function Type (PFT) concept from an original CLM version (i.e., CLM_45_10). As shown in Figure 5, the visualization of related functional groups between two versions of CLM clearly illustrates the model structure changes and help model understand the internal connectivity between modules. The left panel (Figure 5A) shows the submodel structure (including CanopyFluxes and Photosynthesis), while the right panel (Figure 5B) shows the internal connectivity between these functional modules with new additions of ED component (e.g., Photosynthesis_ED and AccumulatedFluxes_ED).

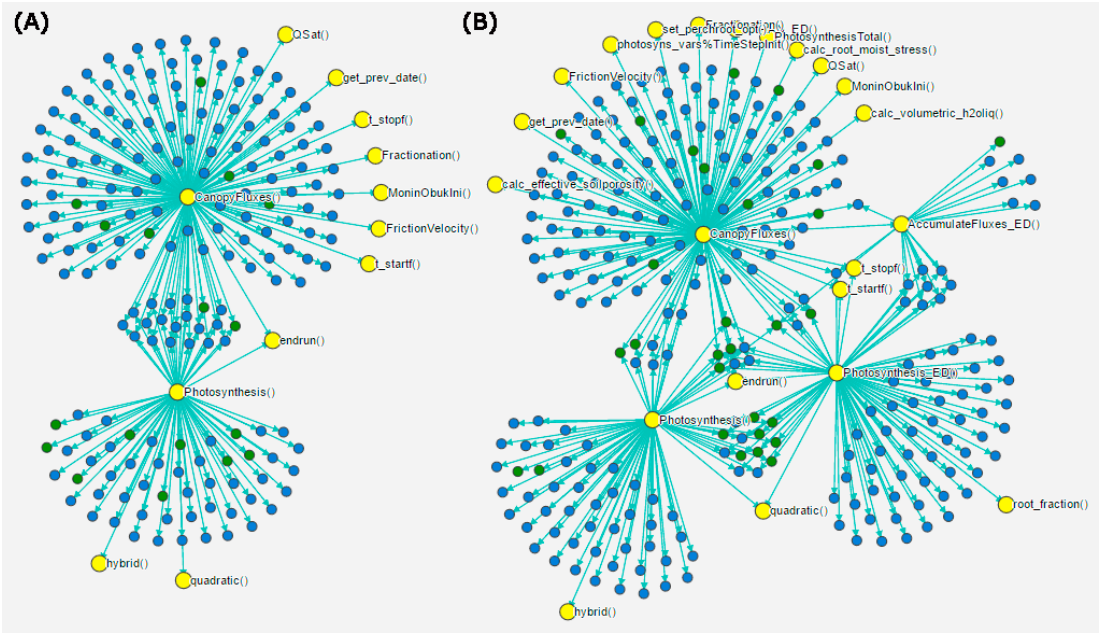

Figure 5: (A) The submodel structure of CanopyFluxes and Photosynthesis in CLM_45_10; (B) Some new subroutines, such as AccumulatedFluxes_ED and PhotoSynthesis_ED, are incorporated into ACME_CLM_V0 as an alternative submodel for photosynthesis representation.

5 Discussion and Conclusion

We present a web-based visual analytic framework to better understand the software structures of large-scale environmental models. The framework adopts a three-tier architecture that integrates data storage, software structure analysis and web-based visualizations. The framework 1) provides an efficient way to manage and analyze the software structure of large scale environmental models, 2) allows users to investigate the environmental model structures with web browser. The system also provides several functions (software structure overview, relational query of functions/variables and visualization of submodel structures) that help users gain insights of environmental model structures from different perspectives and across scales.

A web-based visual analytic system for CLM is located at (`http://cem-base.ornl.gov/CLM_Web/CLM_Web.html`). This system is being used by CLM modelers and field scientists to tackle different environmental research problems. Modelers that are relatively new to CLM can quickly understand the software structure and mechanism by using our web application. Experienced modelers can also use the visualization functions to investigate individual ecosystem processes and how they interact with each other in a particular modeling context. For example, the system was used by researchers to identify key functions and variables that are tied to root-related processes in CLM [9]. These model components were then used to build a new virtual ecosystem dynamic model to assist the exploration of root-related processes.

Currently the system can be used effectively to support dynamic queries and visualizations of software structures from a few perspectives. There are other analytical capabilities that could be incorporated into the system to yield additional insights into the models software structures. For example, how to track the changes between two different software versions and

highlight the difference of their use of variables/functions is important to the understanding of model evolution. It would also be meaningful to incorporate advanced software profiling and debugging tools to gain insights into the computational performance of individual ecosystem functions. These new analytical capabilities could benefit development-related decisions, and provide clues about how the models can be better designed in the future to tackle challenging environmental research questions.

6 Acknowledgement

The majority of this research was funded by the Biological and Environmental Research, Accelerated Climate Modeling for Energy and Terrestrial Ecosystem Science projects, US. Department of Energy. Oak Ridge National Laboratory is managed by UT-Battelle LLC for the Department of Energy under contract DE-AC05-00OR22725.

References

[1] M Luke McCormack, Elizabeth Crisfield, Brett Raczka, Frank Schnekenburger, David M Eissenstat, and Erica AH Smithwick. Sensitivity of four ecological models to adjustments in fine root turnover rate. *Ecological Modelling*, 297:107–117, 2015.

[2] Hausi A Müller, Mehmet A Orgun, Scott R Tilley, and James S Uhl. A reverse-engineering approach to subsystem structure identification. *Journal of Software: Evolution and Process*, 5(4):181–204, 1993.

[3] Keith W Oleson, David M Lawrence, B Gordon, Mark G Flanner, Erik Kluzek, J Peter, Samuel Levis, Sean C Swenson, E Thornton, Johannes Feddema, et al. Technical description of version 4.0 of the community land model (clm). 2010.

[4] M-AD Storey and Hausi A Muller. Manipulating and documenting software structures using shrimp views. In *Software Maintenance, 1995. Proceedings., International Conference on*, pages 275–284. IEEE, 1995.

[5] D Wang, W Wu, T Janjusic, Y Xu, C Iversen, P Thornton, and M Krassovisk. Scientific functional testing platform for environmental models: An application to community land model. In *International Workshop on Software Engineering for High Performance Computing in Science, 37th International Conference on Software Engineering*, 2015.

[6] Dali Wang, Wilfred M Post, and Bruce E Wilson. Climate change modeling: Computational opportunities and challenges. *Computing in Science & Engineering*, 13(5):36–42, 2011.

[7] Dali Wang, Joseph Schuchart, Tomislav Janjusic, Frank Winkler, Yang Xu, and Christos Kartsaklis. Toward better understanding of the community land model within the earth system modeling framework. *Procedia Computer Science*, 29:1515–1524, 2014.

[8] Dali Wang, Yang Xu, Peter Thornton, Anthony King, Chad Steed, Lianhong Gu, and Joseph Schuchart. A functional test platform for the community land model. *Environmental Modelling & Software*, 55:25–31, 2014.

[9] Yang Xu, Dali Wang, Colleen M Iversen, Anthony Walker, and Jeff Warren. Building a virtual ecosystem dynamic model for root research. *Environmental Modelling & Software*, 89:97–105, 2017.

[10] Yang Xu, Dali Wang, Tomislav Janjusic, and Xiaofeng Xu. A web-based visual analytic system for understanding the structure of community land model. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.