Environmental Modelling & Software 89 (2017) 97–105

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Short communication

Building a Virtual Ecosystem Dynamic Model for Root Research

Yang Xu [a, c], Dali Wang [b, *], Colleen M. Iversen [b], Anthony Walker [b], Jeff Warren [b]

[a] Department of Geography, University of Tennessee, Knoxville, TN, USA
[b] Environmental Science Division, Climate Change Science Institute, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[c] Senseable City Laboratory, SMART Centre, Singapore 138602, Singapore

ARTICLE INFO

Article history:
Received 15 December 2015
Received in revised form
30 June 2016
Accepted 11 November 2016

Keywords:
Functional test framework
Community Land Model
Root function
Ecosystem processes

ABSTRACT

Understanding the fundamental mechanistic processes within large environmental models has great implications in model interpretation and future improvement. However, obtaining a good understanding of these processes can be challenging due to the complexities in model structures and software configurations. This paper introduces a functional test framework - with unique approaches to tackling software complexities in large environmental models – to facilitate process-based model exploration and validation. A Virtual Ecosystem Dynamic Model is developed as a case study to better understand and validate root-related processes in the Community Land Model (CLM). The proposed framework could help empiricists better access the inner workings of large environmental models, and facilitate integrative collaborations among broad scientific communities including field scientists, environmental system modelers, and computer scientists.

Published by Elsevier Ltd.

1. Introduction

Over the past several decades, many computer models have been developed to examine numerous mechanistic processes of environmental systems to better predict the future of our natural and built environment. With the rapid development of computing technologies, many high performance and integrated environmental models have been proposed to tackle novel research challenges (Arnold, 2013; Bergez et al., 2013; Granell et al., 2013). These large-scale and integrated models have advanced our understanding of environmental systems. However, software complexities quickly become an issue that hinders model interpretation and future improvement. Such complexities are partially reflected by the intricate relations among various model components. Also, the wide adoption of numerous scientific and parallel libraries introduces additional challenges in software configurations, which limit the applicability of these models for general use.

As a result, it becomes difficult for researchers with varying scientific expertise and technical background to take full advantage of these models, especially when their research focuses on particular ecosystem processes represented across multiple model components. Hence, we need new tools to eliminate or at least mitigate these software complexities to facilitate process-based model exploration and validation. This would inspire integrative collaborations among broad scientific communities that would lead to new model insights and improvements.

In this study, we present an approach to tackling software complexities in large environmental models to provide convenient ways for process-based model exploration. Our previous efforts demonstrated proof of principle for the functional test platform for the examination of internal ecosystem processes in large environmental models (Wang et al., 2014, 2015). In those studies, we performed initial tests of the platform on the highly refined and validated model framework devoted to key leaf photosynthetic processes. In contrast, this current work further develops the functional test framework and applies it to the set of root-related processes that are neither refined, nor well validated in most large environmental models (Warren et al., 2015). Unlike previous studies which examine models' software systems from a topological perspective (Myers, 2003; Zhang et al., 2010), our proposed framework values the hypotheses, scientific workflow, and numerical methods inherited from existing model development. The framework makes it possible for empiricists, such as environmental scientists to focus on the fundamental processes tied to their own research interests without worrying about the complexities in model structures and software configurations. To demonstrate the capability of the framework, we use the Community Land Model

* Corresponding author.
E-mail addresses: yxu30@vols.utk.edu, yangxu@smart.mit.edu (Y. Xu), wangd@ornl.gov (D. Wang), iversencm@ornl.gov (C.M. Iversen), alp@ornl.gov (A. Walker), warrenjm@ornl.gov (J. Warren).

http://dx.doi.org/10.1016/j.envsoft.2016.11.014
1364-8152/Published by Elsevier Ltd.

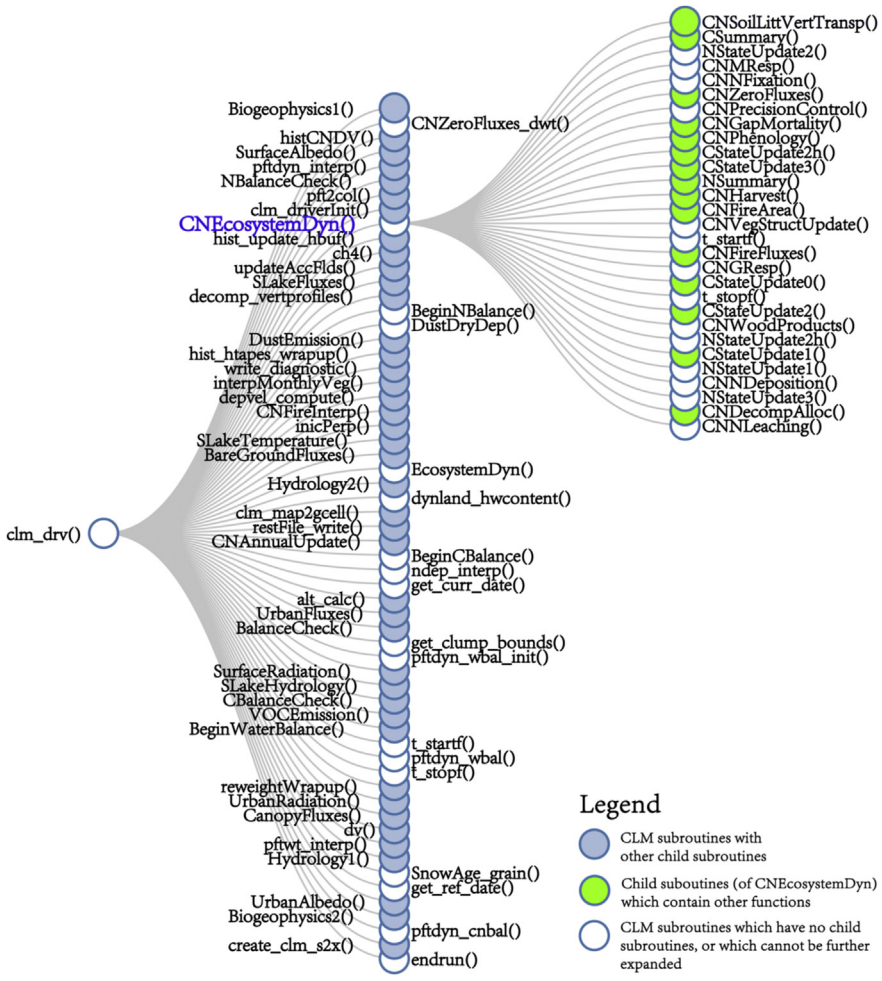

Fig. 1. Software structure of CLM represented by a call tree.[1] The main driver of CLM (i.e., clm_drv) consists of numerous subroutines related to land biogeophysics, biogeochemistry, hydrological cycle, human dimension, and ecosystem dynamics (blue nodes denote the subroutines that contain other child subroutines). Within CLM, there is one subroutine called *CNEcosystemDyn*, which is tied to the root-related processes in the model. As illustrated in the graph, the child nodes of *CNEcosystemDyn* represent the subroutines that simulate vegetation dynamics including phenology, composition, structure, carbon-nitrogen allocation, vegetation respiration, etc. (Green nodes denote the subroutines which contain other functions). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(Oleson et al., 2010) as a case study, and introduce our experience of generating a Virtual Ecosystem Dynamic Model to assist the exploration of root-related processes. We believe our effort to establish the functional test framework could introduce new opportunities for process-based multiscale model verification and validation, and benefit other research programs which face similar challenges in large-scale environmental modelling.

2. The Community Land Model and need for a Virtual Ecosystem Dynamic Model

The Community Land Model (CLM) is the land surface component within the Community Earth System Model (CESM, http://www2.cesm.ucar.edu). CLM is designed to simulate the interaction between terrestrial ecosystems and the climate system in response to natural and human perturbation (Bonan, 1998; Dickinson et al., 2006; Oleson et al., 2010). CLM contains numerous subroutines related to land biogeophysics, biogeochemistry, hydrological cycle, human dimension, and ecosystem dynamics (Fig. 1). Each subroutine is organized by other child subroutines based on natural system functions. These subroutines serve as interrelated components in CLM by interacting with variables that are globally accessible or subroutine explicit (Fig. 2). The software system of CLM adopts many external numerical libraries and parallel computing technologies in order to enhance the model's computing performance. However, the software overhead and the complexities of model structure become a barrier that hinders the assessment of individual subroutines or processes and how they might be improved in future models.

Recently, there has been a great demand to improve root representations within large environmental models (Iversen, 2014; McCormack et al., 2014; Warren et al., 2015). The narrow-diameter, short-lived, fine roots of vascular plants - a belowground analog of leaves responsible for plant water and nutrient acquisition - contribute disproportionately to ecosystem carbon, water, and energy fluxes (McCormack et al., 2015a). However, the distributions and functional dynamics of fine roots from ecosystems spanning the globe are poorly resolved in terrestrial biosphere models such as CLM (Warren et al., 2015). This has led to a disconnection between belowground process data collected by

[1] Figs. 1 and 2 were retrieved from a website which is developed by the authors to visualize the software structures of CLM. For more information, please refer to the website (http://cem-base.ornl.gov/CLM_Web/CLM_Web.html) or relevant articles (Xu et al., 2014, 2016).

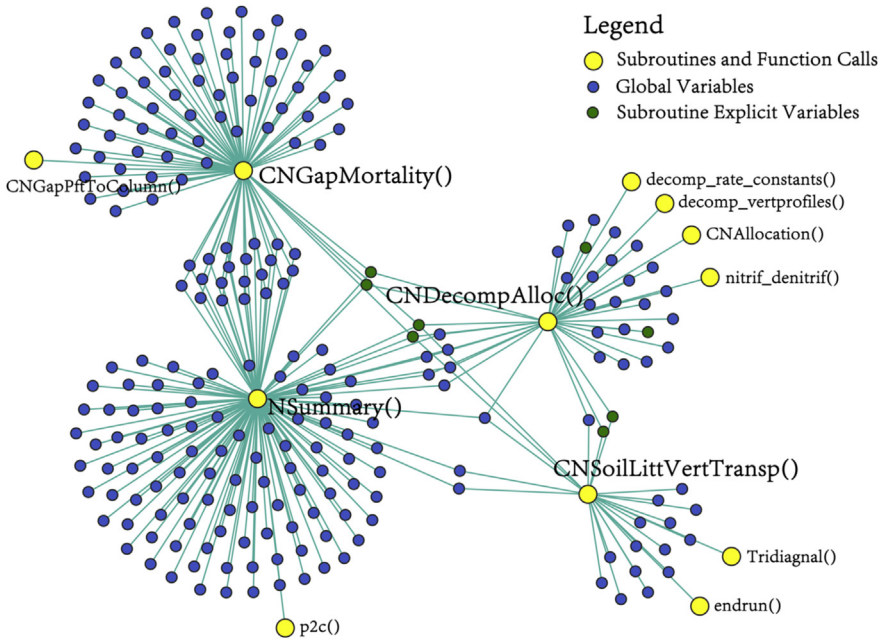

Fig. 2. The interactions among subroutines and variables within *CNEcosystemDyn*. For demonstration purpose, only a selected number of subroutines and related variables/functions are shown. The yellow nodes in the graph denote subroutines and functions that are executed within *CNEcosystemDyn*. The blue nodes denote the global variables, and the green nodes represent subroutine explicit variables. The links between the nodes describe how subroutines and variables access or are accessed by other components. In *CNEcosystemDyn*, its child subroutines (e.g., *CNSoilLittVertTransp*, *CNGapMortality*, *NSummary* and *CNDecompAlloc*) are executed in a particular sequence, and certain variables are accessed or modified by multiple subroutines during the module execution. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

empiricists and adequate model representations of those key root processes, with implications for the accuracy of model projections of ecosystem carbon, water, and nutrient cycling (Iversen, 2010; McCormack et al., 2015b). This disconnection is due in part to limited communication among empiricists and modelers, although these relationships continue to steadily improve (e.g., Medlyn et al., 2015). Empiricists also have limited access to the inner workings of terrestrial biosphere models, and therefore a poor understanding of model representation of root functions, or their responses to environmental conditions (Matamala and Stover, 2013). Hence, new frameworks are needed to provide a means through which root and rhizosphere ecologists can interface with a portion of a terrestrial biosphere model, and conduct model experiments or uncertainty analyses directly focused on their interests. A novel interface could re-align model and empirical experimental research by guiding new measurements and inspiring empiricists to contribute collected data directly to databases for modelling applications.

Within CLM, there is a *CNEcosystemDyn* module (Fig. 1), which simulates vegetation dynamics such as phenology, structure, carbon-nitrogen allocation and composition, and respiration. The module explicitly includes the essential root functions related to carbon and nitrogen cycling. Therefore, it is desirable to develop a standalone ecosystem dynamics model using the built-in functions from CLM to: (1) engage both modelers and empiricists in process-based assessment of model function via an approachable computational experimental setup; (2) enable model-experiment comparison to direct further model improvements, and (3) expedite model structural enhancement to incorporate new knowledge from empirical observations and field experiments. Since the standalone model will contain all the key functions from the ecosystem dynamics module (*CNEcosystemDyn*) within CLM, and have to be integrated with a specific functional testing framework, we name the standalone model as Virtual Ecosystem Dynamic Model, and use it to better understand and validate root-related processes.

3. Methodology and key components

3.1. System architecture of functional test framework

The functional test framework is a software design that allows modelers to test particular ecosystem functions in an environmental model. Fig. 3 shows the system architecture and workflow of the framework using CLM as an example. In this workflow, we first applied several software engineering approaches to generate a functional test module (e.g., *CNEcosystemDyn*) from the corresponding subroutine in CLM. The generated module serves as a standalone unit and can be launched by a functional test driver on users' individual workstations. Next, we executed the original CLM model and during the model simulation, the input and output data streams of the corresponding CLM subroutine (i.e., *CNEcosystemDyn*) was recorded. Then, a data dependency analysis was performed using the input and output data streams extracted from the original CLM simulation. The purpose of the data dependency analysis is to guide variable initialization in order to drive the functional test module in a multi-time-step simulation. Finally, a functional test platform was built that integrates a user interface, functional test driver, and data visualization to allow users to customize the simulation processes and explore the modelling results.

3.2. Functional test module generation

To generate a functional test module, the first step is to reduce the software dependency on parallel computing and external libraries. As Fig. 4 illustrates, we applied several methods including: (1) use of a sequential version of MPI (Message Passing Interface), (2) reconfiguration of the parallel IO library to enable sequential IO access, and (3) installation of only a minimal subset of external

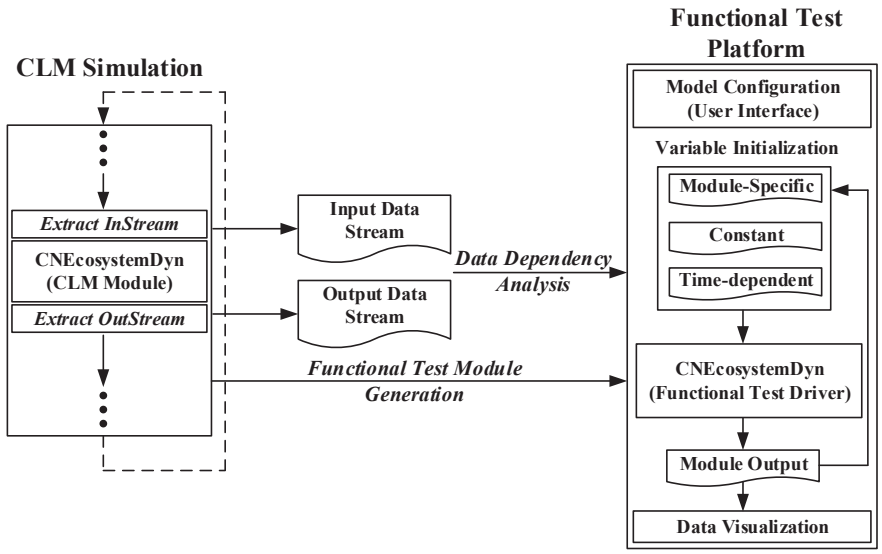

Fig. 3. System architecture and workflow of the functional test framework (using *CNEcosystemDyn* as an example).

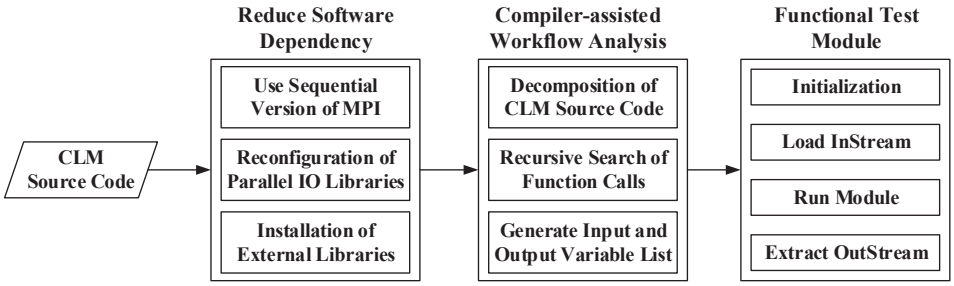

Fig. 4. Work flow of functional test module generation.

libraries, to generate a sequential version of CLM. Due to internal biogeophysical and geochemical connections and software design reasons, CLM simulations have to be executed within CESM with other earth system components. In the software reconfiguration, we removed several external libraries (e.g., MPI, NetCDF, PIO and Coupler) from the original source code using proxy libraries or components. But the key data structure used by CLM (e.g., clm_type) was still kept as the same as before. Hence, data used for the original CLM simulation can be directly used in the functional test module.

Next, a compiler-assisted workflow analysis was performed to better understand the internal data structure and scientific workflow of CLM subroutines. For a given functional test module, we applied a programming language parser tool to capture the input and output data streams of the corresponding subroutine from the original CLM simulation. The tool deconstructed the CLM source code into identifiable tokens (i.e., function calls and variables). During the scanning process, the tool recorded the name of these tokens along with the information on how they access or are accessed by other CLM function calls or variables. The purpose of this step is to extract all the variables that are needed to drive the functional test module.

The test module usually involves four components in a single-time-step simulation: *Initialization*, *Load InStream*, *Run Module*, and *Extract OutStream*. The *Initialization* component prepares the initialization functions for the test module. *Load InStream* contains a subroutine that loads the input variables to drive the test module. *Run Module* contains an execution call to the test module. *Extract OutStream* contains a subroutine that writes the model results into output.

3.3. Data dependency analysis for functional test module

Although the software dependency of the functional test module has been eliminated, the generated test module cannot be directly used for experiments which usually require multi-time-step simulations. The reason is that the output generated by a functional test module at a particular time step does not include all the information to drive the module at the next time step. This creates problems for variable initialization as the functional test module is separated from other modules (i.e., subroutines) in CLM. To tackle this issue, we performed a data dependency analysis for the functional test module by analyzing the relationship between its input and output data streams from the original CLM simulation. The purpose was to group the input variables into several categories based on their unique characteristics. The way the input variables were categorized served as critical information for variable initialization in a multi-time-step simulation.

Fig. 5 illustrates the workflow of the data dependency analysis. We first generated the input and output data streams of the corresponding subroutine from the original CLM simulation for a given period of time (e.g., one year). The simulation period can be determined by the users based on experimental or study objectives. We then analyzed the relationship between module output at each time step x and module input at the next time step $x+1$. By analyzing their relationships, we categorized the input variables of

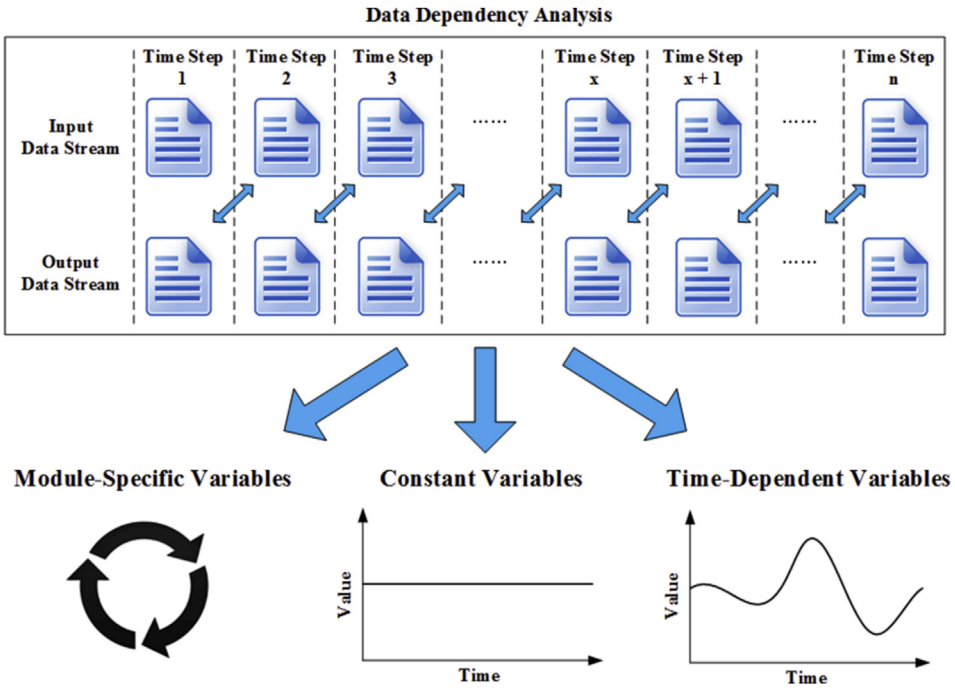

Fig. 5. Data dependency analysis for a given functional test module. By analyzing the relationship between the input and output data streams from the original model (CLM) simulation, the input variables which are needed to drive the functional test module (in a multi-time-step simulation) are categorized into three groups: (1) module-specific variable; (2) constant variables, and (3) time-dependent variables. The three categories are then used to guide variable initialization for the functional test module.

the functional test module into three main categories: (1) module-specific variables, (2) constant variables, and (3) time-dependent variables.

Module-specific variables are the input variables that can be directly retrieved from module output. These variables appear in both input and output of the functional test module, and the value of each variable in the output at time step x remains the same in the module input at time step $x + 1$. Module-specific variables are not modified by other subroutines in the original CLM simulation. Hence, they only need to be initialized at the very first time step in a multi-time-step simulation.

Constant and time-dependent variables are the input variables whose values need to be retrieved from other subroutines. Hence, their values need to be provided at each time step in the simulation process. The constant variables refer to the ones whose values keep constant during the simulation period. The time-dependent variables refer to the ones whose values change over time through the simulation. Thus, constant variables need to be initialized only once in a multi-time-step simulation, while the values of time-dependent variables need to be provided at the beginning of each individual time step.

3.4. Functional test platform

The functional test platform integrates a user interface, a functional test driver, and a data visualization function (Fig. 3). The user interface allows users to set the input variables by the three categories. The functional test module is launched once every time step

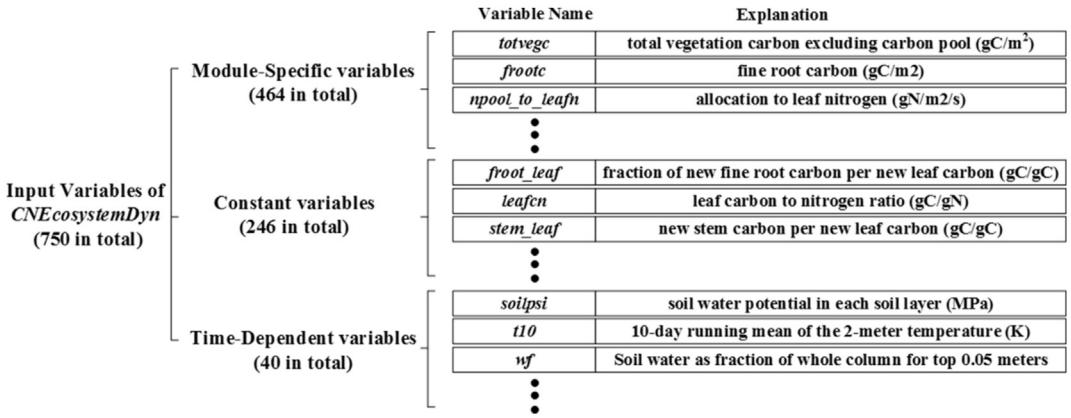

Fig. 6. Results of data dependency analysis for the *CNEcosystemDyn* module. The input variables (750 in total) were categorized into 464 module-specific variables, 246 constant variables, and 40 time-dependent variables. Examples of module-specific variables include *totvegc*, *frootc* and *npool_to_leafn*, which are constantly updated by *CNEcosystemDyn* in the modelling process in CLM. Examples of constant variables include *froot_leaf*, *leafcn* and *stem-leaf*, which describe the physiological characteristics of different vegetation types. Examples of time-dependent variables include *soilpsi*, *t10* and *wf*, which describe external environmental conditions that change over time.

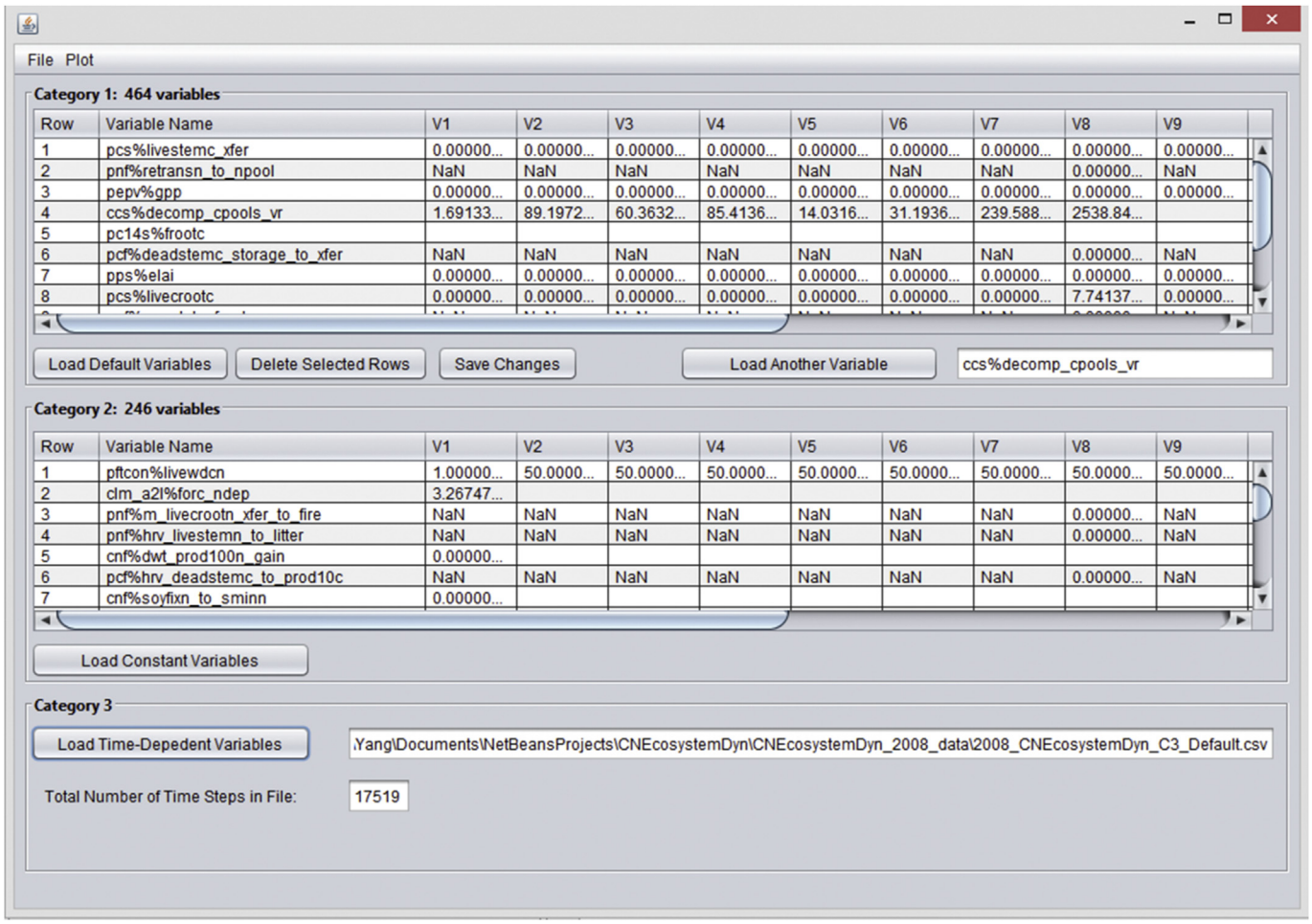

Fig. 7. Main user interface for the *CNEcosysemDyn* functional test module. "Category 1" refers to module-specific variables. "Category 2" refers to constant variables, and "Category 3" denotes time-dependent variables. Columns such as v1, v2, v3, etc., are used to represent the dimensions (e.g., vegetation types) of the variables. The variables are loaded through external files prepared by the users. For Category 3, the number of columns in the external file equals the number of time-dependent variables (40 in this case study), and the number of rows equals the number of total time steps for the simulation (17,519 in this case study[2]). The "Plot" function allows users to visualize the modelling results.

and the input variables are updated at the beginning of each time step as required for each type of input variable. A multi-time-step simulation generally works as follows: (1) At the very first time step, the platform integrates module-specific, constant and time-dependent variables, and organizes them as the input to drive the functional test module; (2) The functional test module is then launched to generate a module output; (3) The platform extracts the values of module-specific variables from the output, and updates the values of these variables for input at the next time step. The values of time-dependent variables are also updated based on the input data stream provided by the user; (4) The functional test module is launched using the updated module input to generate new output for the next time step; (5) The simulation process iterates until it reaches the total number of time steps defined by the user. During the simulation process, the platform stores the module output for each time step, and provides a GUI tool for data visualization.

4. Case study: CNEcosystemDyn

4.1. Data dependency analysis and module validation

We first performed the data dependency analysis on *CNEcosysemDyn* module using the input/output data streams from original CLM simulation for a predefined period of time (one year in this case study). As shown in Fig. 6, we categorized the input variables (750 in total) into three groups, which consist of 464 module-specific variables, 246 constant variables, and 40 time dependent variables. The data dependency analysis benefits modelers and field scientists in several ways. On one hand, modelers could better assess the role of each input variable in *CNEcosysemDyn* based on its derived category. The results help them better understand the hypotheses embedded in the model, and assist variable initialization for experimental designs. On the other hand, the results could engage field scientists by suggesting what variables (e.g., certain constant and time-dependent variables that describe external environmental conditions) need to be measured in order to facilitate model-data comparison.

To assist users in running the functional test module, we developed a user interface for *CNEcosystemDyn*. The interface allows users to configure the module input by uploading external files (e.g., csv format). As Fig. 7 illustrates, the table under "Category 1" stores all the module-specific variables uploaded by a user, and

[2] In this case study, we exported the input and output data stream for *CNEcosystemDyn* subroutine from original CLM simulation for one year (i.e., 2008), which were then used for data dependency analysis and functional test simulation. As each time step in the original CLM equals half an hour, the total number of time steps is thus (24/0.5)*365 = 17,520. We removed the first time step in our analysis because many variables were initialized after the first time step. Thus the total number of time steps for the time-dependent variables is 17,519 in our case study.

Table 1
Experimental design.

Experiment 1		Experiment 2	
Variable	*froot_leaf*	**Variable**	*leafcn*
Description	Fraction of new fine root carbon (C) per new leaf carbon (unit: gC/gC)	**Description**	Leaf carbon (C) to nitrogen (N) ratio (unit: gC/gN)
Values	*froot_leaf* = 0.21	**Values**	*leafcn* = 10
	froot_leaf = 0.42 (default)		*leafcn* = 30.69 (default)
	froot_leaf = 0.84		*leafcn* = 60

the table under "Category 2" stores all the constant variables. "Category 3" denotes the time-dependent variables, which are organized as time-series data in the external file.

We next verified that the *CNEcosystemDyn* module could reproduce the modelling results of the corresponding subroutine in CLM. In particular, we prepared the three types of input variables using the default values extracted from the original CLM simulation. The input variables were then used to drive the functional test module (i.e., *CNEcosystemDyn*) in a multi-time-step simulation. Then, we examined the values of the output variables generated by the functional test module and the corresponding subroutine from the original CLM simulation. By comparing the two sets of output, we found that the functional test module could replicate the results of the original CLM simulation at each particular time step (with a matching rate of 100 percent). Once the result from the functional test module has been validated using the default values from CLM simulation, the functional test module can be used to investigate various model responses and sensitivities to the input variables in the three categories.

4.2. Example application at ORNL FACE

As an example application of the *CNEcosystemDyn* functional test module, we applied the module to simulations of the Oak Ridge National Laboratory Free Air CO_2 Enrichment (ORNL FACE) experiment. We used a simulation of CLM4.5 applied to the ambient CO_2 treatment ORNL FACE following the method and protocol of Walker et al. (2014) and Norby et al. (2015). For demonstration purposes, we tested the module by modifying two plant functional type (PFT) parameters (i.e., Category 2), *leafcn* and *froot_leaf*, and investigated their effect on root related carbon and nitrogen processes. Table 1 shows the experimental design. For each experiment, we set three different values (with one as the default value from benchmark case in the original CLM simulation) to the corresponding input variable and then compare the modelling results.[3]

In each experiment, we compared the benchmark case with the other two cases (test cases) to discover which output variables were most sensitive to each input constant variable. The purpose was to explore the effect of the input variable on the modelling result. For each output variable, we measured the maximum relative difference (RD_{max}) and normalized root-mean square deviation (*NRMSD*) between the benchmark case and each of the other two cases. For each output variable, given a benchmark case $Y = \{y_1, y_1, \ldots y_n\}$ and a test case $\hat{Y} = \{\hat{y}_1, \hat{y}_1, \ldots \hat{y}_n\}$, with n being the total number of time steps, the two measures were calculated as follows:

$$RD_{max} = \max\left(\left|\frac{y_1 - \hat{y}_1}{y_1}\right|, \left|\frac{y_2 - \hat{y}_2}{y_2}\right|, \ldots, \left|\frac{y_n - \hat{y}_n}{y_n}\right|\right) \quad (1)$$

$$NRMSD = \frac{\sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}}}{\overline{Y}} \quad (2)$$

For demonstration purpose, we present several key output variables with considerable changes in the two experiments. In the first experiment (Table 2), we notice that changing *froot_leaf* from its default value to 0.21 or 0.84 has a considerable impact on several variables (e.g., *npool_to_frootn_storage*, *cpool_to_frootc_storage*, *frootn_storage*, and *frootc_storage*) that represent the allocation and storage of carbon and nitrogen in the fine roots. However, as illustrated in Table 3, changing *leaf_cn* from its default to 10 or 60 would only have a notable impact on several output variables at the leaf level (e.g., *leafn_to_retransn*, *retransn*, *tempmax_retransn*, *npool_to_leafn_storage*, and *leafn_storage*) rather than the root level. The comparisons in the two experiments help users better assess the impact of certain PFT constants on root-related processes in the module.

4.3. Data visualization

In order to facilitate model comparison, we developed a visualization function in our framework to allow users to compare the output variables among different experiments. A user could choose multiple output files generated by the functional test module with different input settings, and specify the output variable that he/she wants to explore. For example, Fig. 8 shows the comparison result for variable *cpool_to_leafc_storage* in experiment 1 (as discussed in Tables 1 and 2). The result suggests that increasing the value of *froot_leaf* in the simulation would result in an overall increase of *cpool_to_leafc_storage*, which represents the allocation to the carbon storage in the fine roots. The visualization could help users assess model responses during different seasons, and also explore the daily variations by zooming into the plot.

5. Conclusions and future work

This paper presents our effort to build a functional test framework for environmental model development. A Virtual Ecosystem Dynamic Model based on CLM was developed and used as a case study to demonstrate how the proposed framework could assist environmental scientists to better understand particular ecosystem processes (e.g., root-related processes) without worrying about the complexities in software configurations. The functional test framework brings new opportunities to process-based model development. Users could generate functional test modules to understand the inner workings of an environmental model, and customize their experiments by manipulating the three types of input variables (i.e., module-specific, constant and time-dependent variables) based on their own research interests and goals. The

[3] The *CNEcosystemDyn* functional test module is able to simulate root-related processes for a variety of vegetation types (e.g, needleleaf evergreen tree, broadleaf evergreen tree, broadleaf deciduous tree, and so forth. Please see Oleson et al., 2010 for more details). As an example application at ORNL FACE, this study only compared the modelling results for broadleaf deciduous tree.

Table 2
Model comparisons of selected variables in experiment 1 (the '+' and '-' signs describe if RD_{max} are attributed to the increase or decrease of the output variable in the test case as compared to the bench mark case).

Variable Name	Description	Comparison Results			
		froot_leaf = 0.21 vs. *froot_leaf* = 0.42 (default)		*froot_leaf* = 0.84 vs. *froot_leaf* = 0.42 (default)	
		RD_{max}	*NRMSD*	RD_{max}	*NRMSD*
npool_to_frootn_storage	allocation to fine root N storage (unit: gN/m2/s)	46.90 -	1.01	79.04 +	1.71
cpool_to_frootc_storage	allocation to fine root C storage (unit: gC/m2/s)	46.90 -	1.01	79.04 +	1.71
frootn_storage	fine root N storage (unit: gN/m2/s)	25.19 -	0.18	42.47 +	0.31
frootc_storage	fine root C storage (unit: gC/m2/s)	25.19 -	0.18	42.47 +	0.31
cpool_to_leafc_storage	allocation to leaf C storage (unit: gC/m2/s)	6.22 +	0.13	10.48 -	0.23

Table 3
Model comparisons of selected variables in experiment 2.

Variable Name	Description	Comparison Results			
		leafcn = 10 vs. *leafcn* = 30.69 (default)		*leafcn* = 60 vs. *leafcn* = 30.69 (default)	
		RD_{max}	*NRMSD*	RD_{max}	*NRMSD*
leafn_to_retransn	leaf N to retranslocated N pool (unit: gN/m2/s)	536.27 +	30.52	126.51 -	7.2
retransn	plant pool of retranslocated N (unit: gN/m2)	227.19 +	1.07	53.54 -	0.25
tempmax_retransn	temporal annual max of retranslocated N pool (unit: gN/m2)	227.19 +	0.94	6.32 -	0.02
npool_to_leafn_storage	allocation to leaf N storage (unit: gN/m2/s)	207.00 +	4.47	48.83 -	1.05
leafn_storage	leaf N storage (unit: gN/m2)	111.22 +	0.81	26.24 -	0.19

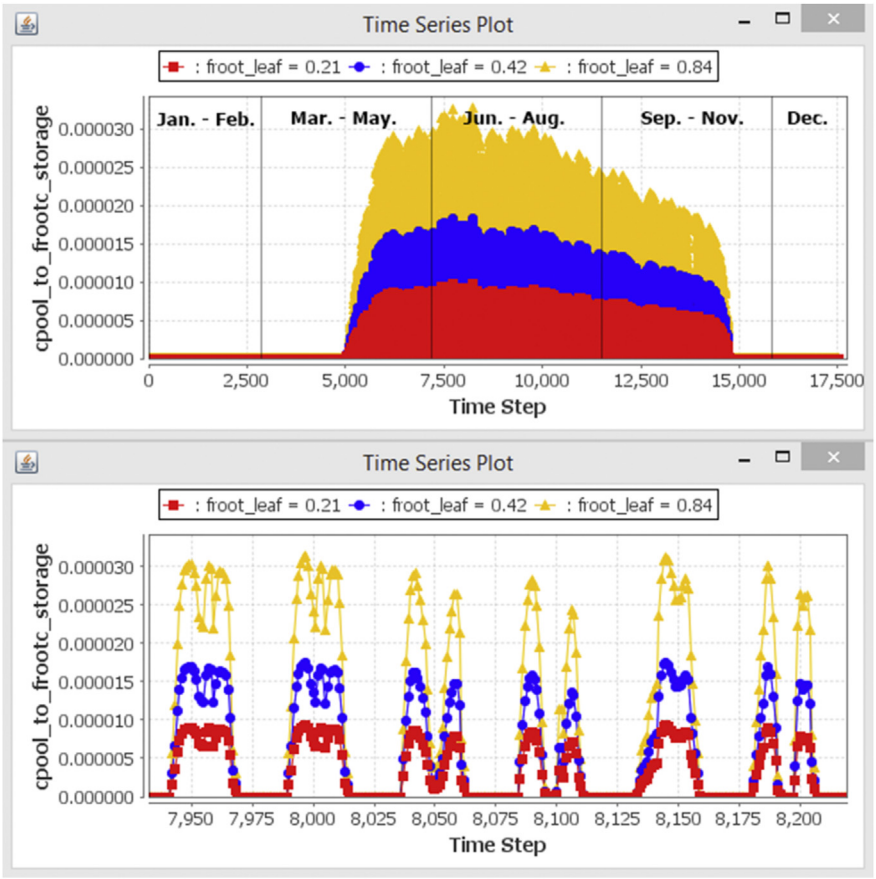

Fig. 8. The comparison of modelling results based on the visualization capability in the functional test framework. The upper figure illustrates the values of an output variable *cpool_to_frootc_storage* based on the three input settings (froot_leaf = 0.21, froot_leaf = 0.42 and froot_leaf = 0.84) in experiment 1 (discussed in Tables 1 and 2). The bottom figure shows the zoom-in part of the top figure. The visualization function enables users to examine the model responses during different seasons (e.g., upper figure), and explore the daily variations (i.e., bottom figure).

customized models can be used to verify and validate internal environmental processes using multiscale observational datasets and field measurement. This would yield valuable insights into model responses, and offer important information for model development and future improvement.

In the future, we plan to broaden our scope by developing a more comprehensive platform that incorporates multiple functional test modules within CLM or other environmental models. The functional test platform will also incorporate many new capabilities: (1) a data synthesis function based on machine learning techniques that allows users to simulate unobserved input variables (e.g., time-dependent variables) based on the observation data at hand; (2) sensitivity analysis for model evaluation and uncertainty quantification; (3) a cloud-based cyberinfrastructure for observational data hosting and model-data comparison. The new platform will serve as a testbed for multi-scale and process-based model exploration and validation.

Acknowledgements

This work was supported by the Office of Biological and Environmental Research in the United States Department of Energy's Office of Science. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy.

References

Arnold, T.R., 2013. Procedural knowledge for integrated modelling: towards the modelling playground. Environ. Model. Softw. 39, 135–148.

Bergez, J.-E., Chabrier, P., Gary, C., Jeuffroy, M.-H., Makowski, D., Quesnel, G., Ramat, E., Raynal, H., Rousse, N., Wallach, D., 2013. An open platform to build, evaluate and simulate integrated models of farming and agro-ecosystems. Environ. Model. Softw. 39, 39–49.

Bonan, G.B., 1998. The land surface climatology of the NCAR land surface model coupled to the NCAR Community Climate Model. J. Clim. 11, 1307–1326.

Dickinson, R.E., Oleson, K.W., Bonan, G., Hoffman, F., Thornton, P., Vertenstein, M., Yang, Z.-L., Zeng, X., 2006. The community land model and its climate statistics as a component of the community climate system model. J. Clim. 19, 2302–2324.

Granell, C., DíAz, L., Schade, S., OstläNder, N., Huerta, J., 2013. Enhancing integrated environmental modelling by designing resource-oriented interfaces. Environ. Model. Softw. 39, 229–246.

Iversen, C.M., 2010. Digging deeper: fine-root responses to rising atmospheric CO_2 concentration in forested ecosystems. New Phytol. 186 (2), 346–357.

Iversen, C.M., 2014. Using root form to improve our understanding of root function. New Phytol. 203 (3), 707–709.

Matamala, R., Stover, D.B., 2013. Introduction to a Virtual Special Issue: modeling the hidden half–the root of our problem. New Phytol. 200 (4), 939–942.

Medlyn, B.E., Zaehle, S., De Kauwe, M.G., Walker, A.P., Dietze, M.C., Hanson, P.J., Hickler, T., Jain, A.K., Luo, Y., Parton, W., Prentice, I.C., Thornton, P.E., Wang, S., Wang, Y.-P., Weng, E., Iversen, C.M., McCarthy, H., Warren, J.M., Oren, R., Norby, R.J., 2015. Using ecosystem experiments to improve vegetation models. Nat. Clim. Change 5, 528–534.

McCormack, M.L., Dickie, I.A., Eissenstat, D.M., Fahey, T.J., Fernandez, C.W., Guo, D., Helmisaari, H.S., Hobbie, E.A., Iversen, C.M., Jackson, R.B., 2015a. Redefining fine roots improves understanding of below-ground contributions to terrestrial biosphere processes. New Phytol. 207, 505–518.

McCormack, M.L., Crisfield, E., Raczka, B., Schnekenburger, F., Eissenstat, D.M., Smithwick, E.A., 2015b. Sensitivity of four ecological models to adjustments in fine root turnover rate. Ecol. Model. 297, 107–117.

McCormack, M.L., Gaines, K.P., Pastore, M., Eissenstat, D.M., 2014. Early season root production in relation to leaf production among six diverse temperate tree species. Plant Soil 389, 121–129.

Myers, C.R., 2003. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. Phys. Rev. E 68 (4), 046116.

Norby, R.J., Oren, R., Boden, T.A., De Kauwe, M.G., Kim, D., Medlyn, B.E., Riggs, J.S., Tharp, M.L., Walker, A.P., Yang, B., Zaehle, S., 2015. Phase 1 free air CO2 enrichment model-data synthesis (FACE-MDS). Meteorol. Data 015. http://dx.doi.org/10.3334/CDIAC/FACE-MDS/MET.01.

Oleson, K.W., Lawrence, D.M., Gordon, B., Flanner, M.G., Kluzek, E., Peter, J., Levis, S., Swenson, S.C., Thornton, E., Feddema, J., 2010. Technical Description of Version 4.0 of the Community Land Model (CLM).

Wang, D., Janjusic, T., Iverson, C., Thornton, P., Karssovski, M., Wu, W., Xu, Y., 2015. A scientific function test framework for modular environmental model development: application to the Community Land Model. In: Proceedings of the 2015 International Workshop on Software Engineering for High Performance Computing in Science. New Jersey, USA, pp. 16–23.

Wang, D., Xu, Y., Thornton, P., King, A., Steed, C., Gu, L., Schuchart, J., 2014. A functional test platform for the Community Land Model. Environ. Model. Softw. 55, 25–31.

Walker, A.P., Hanson, P.J., De Kauwe, M.G., Medlyn, B.E., Zaehle, S., Asao, S., Dietze, M., Hickler, T., Huntingford, C., Iversen, C.M., 2014. Comprehensive ecosystem model-data synthesis using multiple data sets at two temperate forest free-air CO_2 enrichment experiments: model performance at ambient CO2 concentration. J. Geophys. Res. Biogeosci. 119 (5), 937–964.

Warren, J.M., Hanson, P.J., Iversen, C.M., Kumar, J., Walker, A.P., Wullschleger, S.D., 2015. Root structural and functional dynamics in terrestrial biosphere models–evaluation and recommendations. New Phytol. 205 (1), 59–78.

Xu, Y., Wang, D., Janjusic, T., Xu, X., 2014. A web-based visual analytic system for understanding the structure of Community Land Model. In: Proceedings of the 2014 International Conference on Software Engineering Research and Practice.

Xu, Y., Wang, D., Janjusic, T., Wu, W., 2016. A web-based visual analytic framework for understanding large scale environmental models: a case study for the Community Land Model. Comput. Sci. Eng. (under revision).

Zhang, H., Zhao, H., Cai, W., Liu, J., Zhou, W., 2010. Using the k-core decomposition to analyze the static structure of large-scale software systems. J. Supercomput. 53 (2), 352–369.