International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

DLES framework for spatially-explicit simulation modelling: general description and some exemplary applications

Maria Bezrukova, Vladimir Shanin, Yulia Khoraskina, Alexey Mikhailov
Institute of Physico-Chemical and Biological Problems in Soil Science of the Russian Academy of Sciences, 142290 Institutskaya ul., 2, Pushchino, Moscow Region, Russian Federation, shaninvn@gmail.com

Abstract: DLES is a component-based framework integrating forest and soil simulation models with discrete spatial and temporal resolution. The task was to create the framework facilitating combination of stand-alone models of different spatial and temporal resolution with spatial interactions into the unified system, replacing models and comparing models against each other. Also, the feedbacks between these models can be implemented. The main features of the suggested approach are: 1) the system of models isn't a 'monolithic program' (single executable file), but is a number of stand-alone modules that can be easily added to the system and removed from it; 2) modules exchange data with each other using the shared area of memory which is controlled by the special system unit, and therefore, do not have to be concerned with interconnectivity between sub-models; 3) the module can be either sub-model or other data provider (file, database etc.). Some spatially explicit routines are implemented in the system, such as neighbourhood search, automatic iteration on simulation grid, calculation of distance between objects on this grid, and disposal of the edge effect. The customization of the system of models is being attained with the scheme of the system of models.

***Keywords**:* component-based software, frameworks, environmental modelling, model integration

1 INTRODUCTION

As ecosystem processes have a very wide range of spatial and temporal scales, it is possible to develop a great amount of models simulating these processes. The development of a simulation model is the quest about compromise between truth to nature of simulated object and simplification of its description. On one hand, it is to be wished that model will be able to represent as much as possible characteristics of object. However, the amount of necessary input data increases with an increase of model refinement.

Traditional approach is the development of 'monolithic' programs with source code compiled into single executable file. It does not allow creating flexible system of models. It was noted by He et al. [2002] that solution of such problem can be achieved by so-called 'modular-based' applications. Modularity is one of principles of system constructing when functionally-related parts are being separated into modules. These modules can further be united into the whole system in runtime, i.e. after program start. Modular design has some advantages. For example, the development and modification of individual component is much easier task than development of the whole system of models. Different modules of the system can be written by different research groups working independently and using different programming languages and approaches to software development. The modules

of complex software systems can be easily replaced by others without full re-compilation. It is also possible to create system changing its own functionality according to specified requirements.
The purpose of this article is to describe structure, functioning and basic features of the framework for building of component-based systems of simulation models. More details are reported by Bezrukova et al. [2012, *in press*].

2 DLES: STRUCTURE, FUNCTIONING, FEATURES

The main objective was to create a framework which can be used as the basis of assemblage of individual-based simulation models with discrete spatial and temporal resolution. However, DLES (Discrete Lattice Ecosystem Simulator) wasn't developed as an all-purpose system. We wanted to create a framework which will facilitate the implementation of some routines arising during the development of models. So the model should be reconstructed into standardized model components, pieced together to form a new system of models with the desired characteristics. The development of the system of models can be greatly facilitated by using libraries of reusable model components. The system presented in this paper has its own niche between systems of models based on 'monolithic' concept (the model developer should be a programmer to write the whole system) and specialized mathematical frameworks with a set of pre-implemented methods and procedures, such as MATLAB-Simulink (see Gray [2011]), Wolfram Mathematica (see Wolfram [2003]), Maple (see Lynch [2010]), and others. Main requirements to the DLES structure and interface have been defined of many years of applications of the EFIMOD system of simulation models of forest growth and elements cycles in forest ecosystems [Komarov et al., 2003].

2.1 Structure

Advantages and disadvantages of the existing implementations of component-based systems were taken into account while developing DLES. It is focused on the simulation of complex ecological systems with discrete spatial and temporal step. DLES allows taking into account some specific problems coming from development of individual-based forest stand models, such as programming of local interactions between trees or matching of spatial and temporal scales. The simulation is carrying out on the two-dimensional grid with Cartesian coordinate system. This grid is divided into cells. This simplifies the development of models with localized objects (e.g. trees) by means of more simple and fast algorithms of searching of neighbours to describe the interactions between objects because both distance between trees and each tree's position on simulation grid can be expressed in terms of cells instead of using the distance measurement units. Such structure also simplifies the representation of spatial discontinuity.
Any DLES module (called 'component') is the functionally completed and self-sufficient system unit. The system *kernel* is the main component. It provides interaction between components and their 'team-work' through performance of control and supporting of the data exchange. *Shell* provides user interface, i.e. it transfers user commands to the system and receives the results of modelling to display. *Models* (*sub-models*) are the implementations of algorithms which simulate certain processes in the ecosystem. Sub-models receive initial data and return the results of simulation. Files with input data also can be represented as sub-models.
Each component is implemented as a dynamically linked library (DLL) or executable file (EXE) which contains operating algorithm implemented in machine-language code. DLES uses COM-interfaces (see Rogerson [1997]) to link components so that these components can be written on different programming languages. At the current stage, we use for DLES this rather old technology instead of more modern and powerful .NET because COM copes all problems which we faced and don't requires installation of additional software like .NET Framework for software execution. However, these technologies are compatible,

which will further provide us with smooth migration form COM to .NET. Thus, the *system of models* is a software system of components and links between them.
Kernel integrates all components into the unified system using information that is enclosed in the *scheme* of the system of models. The scheme is the description of the system of models, which includes the list of all sub-models, theirs execution order and description of relationships between components. *Scheme editor* is a special program for easy and error-free editing of the scheme of the system of models.
It is easy to create a new sub-model for DLES. The developer should make new DLL using any software development environment and add the common unit, which contains interface declarations as well as declarations and implementations of some service routines. Then developer should declare the main class of sub-model (for example, *TModel*). All global variables of sub-model and implementations of the interface must be declared as class members. Sub-model constructor *CreateComp* and method *GetModuleInfo* must be declared as exported function. After that the DLL could be compiled and used in the DLES.
The basic concept of DLES is to represent the descriptions of complex ecological processes as the assemblage of elementary ones. Each of these processes can be implemented in separate sub-model.
The framework imposes no restriction on the complexity of the sub-model code itself. Sub-models are independent and not directly reference other components; interaction is indirect via updates to state variables.
Each sub-model should provide some information: its name, spatial scale and temporal resolution. Also, the required input variables and available output variables should be listed. The variables that are external and available for other sub-models are called ‘ports’.
Three spatial levels are implemented in the system: cell level, object level, and plot level. They have different execution ways and specific features of memory allocation for their variables. The plot-level sub-models run one time per time step, and the corresponding variables are related to the whole simulation plot. The cell-level sub-models iteratively run for each cell of simulation grid at each step. The variables, which are corresponded to this kind of sub-models, are the two-dimensional data arrays with the size equal to the size of simulation grid. The object-level sub-models are of the third kind. The variables of object-level sub-models are linear arrays. The sub-models of this kind are designed for simulation of the objects which can be more than one on the simulation grid but which have an irregular spatial distribution (for example, trees). Therefore, the storage of such data in the cell-level variables is unpractical. Accordingly, the object-level sub-model iteratively runs for each object on simulation grid during one time step of the system of models.
The spatial levels are hierarchic; the plot-level sub-model can access the whole array of variables of object-level and cell-level sub-models. Plot-level sub-models can also manage own cell- or object-level variables.
Such organization allows a range of spatial-dependent routines, such as searching of the nearest neighbours, estimation of the range between objects on the grid that is required, for example, for computation of degree of competitive interaction between trees. Embedded routines also include cells’ grouping on grounds of neighbourhood (or any other feature) that can be necessary, for example, when determining nutrition zones for each tree. All routines mentioned above will be implemented in the system kernel and will be run on demand.
Sub-models included in the scheme can work with different temporal resolution. To provide the proper ‘team-work’ of sub-models, the system will operate with time step that equal to the minimal temporal resolution among all sub-models.

2.2 Functioning

When all sub-models are ready, it is necessary to combine them into the unified system. To describe such system the scheme is used. This is the XML-document consisting of three sections: (1) general description of the scheme; (2) the description of components; (3) the description of relationships between

components.
The general scheme description contains the characteristics of the simulation grid (its size and size of cells), minimal and maximal numbers of time steps and user comments to the scheme. The description of component contains its name, path to the file from which the component should be loaded, its version, execution order, spatial and temporal resolution. The description of relationship contains relationship type, names of connected ports, and parameters of spatial and temporal synchronization and conversion of measurement units.
All components are combined into the system of models under management of the kernel. At the initial stage, the kernel loads scheme and analyzes it. Then it loads all sub-models listed in the scheme and requests service information (spatial and temporal resolution of sub-model) and information about ports of sub-models.
According to spatial scale and type of variable (numeric, string, Boolean or list), the kernel allocates memory. All ports of all sub-models are put into joint list of state variables managed by the system kernel. At that, each sub-model can access any of these state variables and get theirs values through pointers.
Then the kernel analyses relationships listed in the scheme and connects their ports. As sub-models can work with different spatial and temporal resolution, and the ports of these sub-models may have different units of measurement, it is necessary to provide correct synchronization during data exchange. Therefore, the kernel calculates the coefficients of conversion for each pair of ports. The simplest situation is when it is necessary to convert measurement units. It is possible to convert not only simple (for example, [m] to [km]) but also complex (for example, [t ha^{-1}] to [kg m^{-2}]) measurement units.
Moreover, kernel takes care of spatial and temporal synchronization. For example, let's consider the relationship connecting ports of plot-level sub-model with monthly time step and plot-level sub-model with annual time step. According to the scheme, the plot-level sub-model should be run first. Kernel primarily runs first sub-model twelve times, following which the second sub-model will be run. The method of synchronization can be set by the user and saved in the scheme. For example, this may be computation of the sum for 12 month, or average value. The synchronization between sub-models with different spatial resolution is carried out in the same manner.
At the final stage, the kernel initializes all sub-models, i.e. it sets the initial values for all input variables of sub-models. Now the system is ready to work. On the user's command, the kernel runs the simulation for the given number of steps, successively running all sub-models in the same order as they listed in the scheme.

2.3 Features

DLES user interface is implemented as *Shell* component. Shell translates user commands to the kernel. Among such commands, for example, the initialisation of the system of models or run of the simulation experiment for a given number of steps. It also shows the results of simulation experiment. DLES is a scalable system and one can use different shells depending on the current task.
For investigative and educational tasks, the shell with graphical user-friendly interface (GUI) is the most suitable one. This shell has many possibilities for the visualization of results such as diagrams and maps. Shell with GUI also allows making several simultaneous simulations with different sets of models and/or files with initial data and combining the results of these simulations on one graph plot to make a comparative analysis.
Another type of shell is a console. It is the most suitable one for rapid computations due to high performance. The system of models can be run with command line parameters. For landscape-level modelling, it will be necessary to create a database-oriented shell connected with GIS. This kind of shell will allow making simulations for large territory and showing thematic maps with results.
Often the problem arises when it is necessary to develop the models which must be run by the user only at certain steps instead of working at each step of the system of models (for example, the models of fire events which do not occur at

each step). Moreover, such models must be parameterized by data entered by user directly in runtime. Special class template is provided to facilitate the development of such kind of models. It contains a field which is the sub-model execution flag. Since the shell is allowed to modify the state variables, it is possible to set the initial parameters of 'event' sub-models in runtime.
Specialized tool allows loading data from external sources: text files, datasheets, and databases. It can be represented in the scheme of the system of models as usual sub-model. This tool automatically imports all variables determined by user and represents they as the ports of this sub-model.
Researchers are permanently faced with the necessity of various transformations of results obtained from simulations. Of course, one could make necessary recalculations after the simulation using additional software. However, DLES framework provides an easier way to implement these operations with a special component called *Outputs*. This is a virtual component which not implements any algorithms. The values of all ports of this component are calculated from the values of ports of other sub-models using logical or arithmetic expressions which are determined by user in the scheme of the system of models. The values of expressions are calculated by the kernel in runtime.
DLES allows carrying out simulations not only in 2-dimensional but also in 3-dimensional space. This feature can be implemented with 2-dimensional array of variables of list type where each list represents 3rd dimension.

3 EXEMPLARY APPLICATIONS

DLES allows simplifying routine work, e.g. comparison of different simulation scenarios or different versions of sub-models, site-specific calibration of parameters, analysis of the theoretical basis of sub-models, sensitivity analysis and verification. During the development of such system of models, we will use new possibilities provided by DLES for developing of large simulation model of forest ecosystem. One block of this system of models is ready now. This block simulates the dynamics of organic matter, nitrogen and calcium in soil.
We used new version of, the model of soil organic matter (SOM) dynamics ROMUL. Earlier version was described by Chertov et al. [2001]. It allows calculation of the dynamics of SOM in forest. ROMUL is based on studying of successive stages of mineralization and transformation of fresh litter, which correspond to SOM pools in horizons L, F and H of forest floor and horizons in mineral soil. Rates of transformation can be obtained from the experimental data. Also, the temperature and soil moisture modifiers of the fluxes take a variety of forms given as step-wise defined functions. The optimal conditions for different fluxes are, however, somewhat different. This simple scheme allows for simulating dynamics of other elements of nutrition in soil. At some assumptions, we can link the calcium model with ROMUL through the SOM pools maintaining the same tracks of elements flows but with adding of several pools more. Compounds of calcium form the secondary minerals, which are relatively labile and can transform into exchangeable or available for plants pools of calcium. We assume that rates of decomposition and transformation between elements pools in soil horizons in this case are similar to these ones in ROMUL. Main pools of Ca in SOM are also similar with pools in ROMUL but with some corrections and additions. The model of calcium dynamics was described in details by Khoraskina et al. [2009]; Komarov et al. [2012]. Forest floor and mineral soil water contents and fluxes through forest floor and mineral soil layers were calculated using two-layer water balance model. The general scheme is shown in Figure 1.
Sub-model of soil organic matter dynamics *modSOMSoil* simulates decomposition of litter and SOM in mineral soil (monthly income of litter fall cohorts from different compartments of different tree species from *fLitterFall* file and initial values of SOM pools from *fInitValues* file) in dependence on temperature and moisture of litter and mineral soil (monthly climatic data from *fClimate* file). The sub-model of SOM dynamics is linked with sub-model of calcium dynamics in soil *modCaSoil* through average nitrogen content in litter for calculation of the rates of decomposition of corresponding SOM and Ca pools. *modCaSoil* also obtains monthly data on the

amount of litter fall, initial values of Ca in soil pools and climatic data from corresponding input files. Sub-model *modCaUptake* calculates the beginning of vegetation period where the main driving variable is the sum of active temperatures obtained from *fClimate* file. *fCaUptake* file stores the annual amounts of Ca consumed by plants. *modCaSoil* recalculates these values with annual time resolution to daily/monthly time resolution using the output of *modCaUptake*. The model of soil water balance *modWater* affects the dynamics of leaching of Ca from soil profile in *modCaSoil* through output data on water flows and storages. Input parameters of this model with daily time step are stored in *fWeather* file. File *fInitValues* contains all data required for initialization of sub-models: stocks of soil organic matter, nitrogen and calcium in main pools, their concentrations in litter fall cohorts, and some constants for equations describing rates of decomposition of plant residues in soil, water balance parameters and constants.

As it was mentioned above, DLES can manage sub-models on different levels of spatial organization. We used this feature to simulate spatial inhomogeneity of soil cover. To this end, we ran *modSOMSoil* sub-model on cell-level while other sub-models were executed on plot-level. This allowed simulating the different SOM dynamics in each cell (0.5x0.5 m) of simulation grid. Further, we are going to implement spatially-explicit sub-models of litter fall distribution, ground layer vegetation and distribution of plant roots. These steps could be resulted in remarkable refinement of the model of SOM dynamics because it will account spatial inhomogeneity of litter income, microclimatic conditions and nutrients' uptake by plants. It should be noted here that such a high level of detail, probably, is not needed when simulating some types of habitats such as meadows or agro-ecosystems. In contrast, forest soils have extremely high level of heterogeneity and such model design is rather rational. It is possible to change the structure of the system of models and change one sub-model to another by editing the scheme instead of editing of program code and its further recompilation.

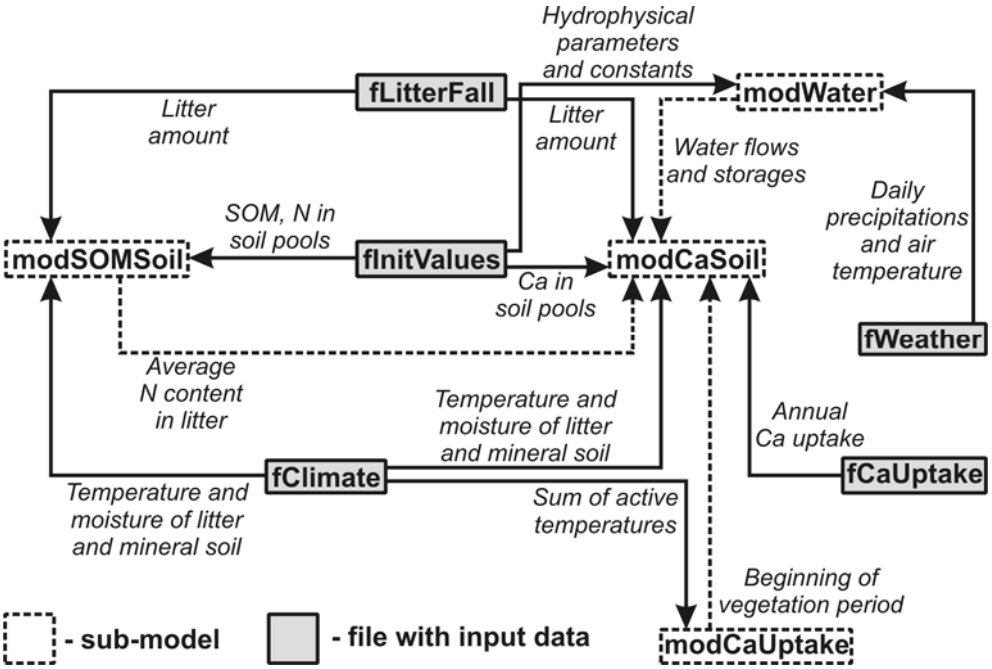

Figure 1. The general scheme of block describing 'soil dynamics'. Simplified representation: only the most important ports connecting components are shown.

This system of models was applied for simulation of Ca dynamics in steady state forest (Khoraskina et al. [2009]). We applied simulation scenario with clear cutting after 10 years of steady state development. In 5 years after cutting the natural regeneration by spruce seedlings was simulated. Pools of Ca in forest floor and exchangeable Ca increased after cutting, in comparison with scenario without cuttings. Ca in forest floor increased in 10 years after regeneration. The pool of

exchangeable Ca had the slower increase. Available Ca (Figure 2, A) had similar dynamics as exchangeable Ca with a peak after cutting. Then it decreased and began to accumulate after 35 years. Ca leaching (Figure 2, B) had similar dynamics, but there were remarkable values of Ca leaching after cutting.

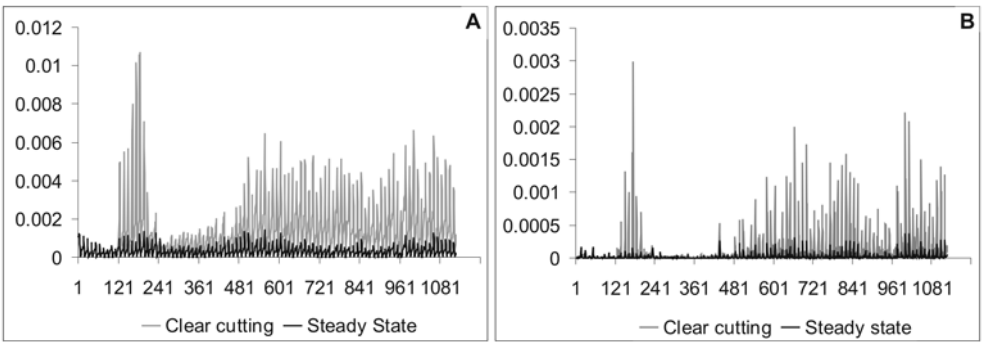

Figure 2. The dynamics of Ca available for plants (A) and Ca leaching (B) in steady state spruce forests in Russian northern taiga. Vertical axis – Ca stock (kg m^{-2}); horizontal axis – simulation step (month).

In this example, we presented the complex system of models. At the first stage of simulation, we calibrated system under the scenario with stable climate for main pools of SOM and calcium in soil using some calibration parameters in *fInitValues* file. DLES was very useful for facilitating and accelerating of the process of calibration because of doing this in real-time with results obtaining immediately. After calibration we simultaneously simulated two projects (steady state scenario and clear cutting scenario) and compared results real-time using the features of DLES. More examples are reported by Bezrukova et al. [2012, *in press*].

4 CONCLUSION

New component-based framework is suggested. It is a scalable system, so it can be used for building of large simulation models of terrestrial ecosystems in different climatic regions taking into account various impacts and silvicultural practice. It allows easy combining of sub-models and provides a range of routines to facilitate accounting of spatial-explicit interactions within sub-models. The system of models within DLES framework has standard description based on human-readable XML. This description includes the list of sub-models and definition of theirs interaction. DLES allows data exchange between sub-models which work with different spatial and temporal resolution. There is also a set of statistical procedures for data processing, comparison of models and sensitivity analysis. When developing DLES, we didn't aim at creating of universal simulation environment. The main task was to develop a tool for combining of spatial-explicit models, i.e. models taking into account the location of simulated objects and distance between them (Goodchild et al. [2002]). DLES is pointed on implementation of discrete models, where an area of simulation divided into cells and location and distance can be expressed in terms these cells. Special attention was paid to modelling of forest ecosystems. This resulted in some specific features, such as support of 3D spatial structures, object-level variables and sub-models, and spatially-dependent routines. No need in full universalization allowed us to create rather simple system in terms of structure and user interface. This approach is different from the all-purposed complex environments such as MATLAB-Simulink, or others. Flexibility is other aspect of this approach. DLES specifies only the protocol of data exchange between different components, allowing arbitrarily complex internal implementation of the sub-models. Convenient and flexible data exchange mechanism is also an important feature of DLES. The individual state variables are the basic unit of data exchange instead of full sets of model outputs. The detailed settings of temporal/spatial scale synchronizing and conversion between different measuring units are possible. The approach used in DLES is the most similar to OpenMI

(Gregersen et al. [2007]) and Capsis (de Coligny [2005]). The most remarkable difference from the OpenMI is a unified user interface and unified management of input data, which are common to all models in the system. Unlike CAPSIS, DLES allows models written with any programming language supporting COM. Changes in the structure of the system of models do not require recompilation of the entire system. In contrast to many other frameworks, kernel unit and user interface in DLES are separated. This allows creating a various independent implementations of GUI for different tasks.

ACKNOWLEDGMENTS

We are grateful to Prof. Alexander Komarov for comments and suggestions. The work was supported by RFBR projects 09-04-01209, 12-04-01269 and Programme of RAS Presidium 'Environment of Russia: adaptations to climate changes and development of nuclear power industry'. We also would like to thank all anonymous reviewers for their insightful comments.

REFERENCES

Bezrukova, M.G., V.N. Shanin, A.V. Mikhailov, N.V. Mikhailova, Yu.S. Khoraskina, P.Ya. Grabarnik, and A.S. Komarov, *DLES – a component-based framework for ecological modelling*. In: Jordán, F., and S.E. Jørgensen, *Models of the Ecological Hierarchy from Molecules to the Ecosphere*. Elsevier, 24 pp., 2012. *In press*.

Chertov, O.G., A.S. Komarov, M.A. Nadporozhskaya, S.S. Bykhovets, and S.L. Zudin, ROMUL – a model of forest soil organic matter dynamics as a substantial tool for forest ecosystem modelling, *Ecological Modelling*, *138*, 289-308, 2001.

de Coligny, F., Capsis: Computer-Aided Projection for Strategies In Silviculture, a software platform for forestry modellers. *Workshop on Information Science for Agriculture and Environment (ISAE). 3-4 june 2005, GuiZhou Normal University, GuiYang, P.R. China*.

Gray, M.A., *Introduction to the Simulation of Dynamics Using Simulink*, Chapman and Hall/CRC, 332 pp., Boca Raton, FL, 2011.

Gregersen, J.B., P.J.A. Gijsbers, and S.J.P. Westen, OpenMI: Open Modelling Interface, *Journal of Hydroinformatics*,. *9(3)*, 175-191, 2007.

Goodchild, M., Issues in spatially explicit modelling, *Agent-Based Models Of Land-Use and Land-Cover Change: Proceedings of an International Workshop October 4–7, 2001, Irvine, California, USA*, Center for the Study of Institutions, Population, and Environmental Change, Indiana University, USA, 2002.

He, H.S., D.R. Larsen, and D.J. Mladenoff, Exploring component-based approaches in forest landscape modelling, *Environmental Modelling and Software*, *17*, 519–529, 2002.

Lynch, S., *Dynamical systems with applications using MAPLE*, 2nd ed., Birkhäuser, 518 pp., Boston, 2010.

Khoraskina, Yu.S., A.S. Komarov, M.G. Bezrukova, N.V. Lukina, and M.A. Orlova, Calcium dynamics model in North taiga forest soils, *Proceedings of Samara scientific centre of Russian Academy of Sciences*, *1(7)*, 1468–1477, 2009. *In Russian*.

Komarov, A.S., O.G. Chertov, S.L. Zudin, M.A. Nadporozhskaya, A.V. Mikhailov, S.S. Bykhovets, E.V. Zudina, and E.V. Zoubkova, EFIMOD 2 – a model of growth and cycling of elements in boreal forest ecosystems, *Ecological Modelling*, *170*, 373–392.

Komarov, A.S., Yu.S. Khoraskina, S.S. Bykhovets, and M.G. Bezrukova, Modeling of soil organic matter and elements of soil nutrition dynamics in mineral and organic forest soils: the ROMUL model expansion, *Procedia Environmental Sciences, 13,* 2012, 525–534.

Rogerson, D., *Inside COM*, Microsoft Press, 416 pp., Redmond, WA, 1997.

Wolfram, S., *The Mathematica Book*, 5th ed., Wolfram Media, 1488 pp., Champaign, IL, 2003.