A PROBLEM-SOLVING WORKBENCH FOR INTERACTIVE SIMULATION OF ECOSYSTEMS

Robert G. Knox, Virginia L. Kalb, and Elissa R. Levine
NASA Goddard Space Flight Center
David J. Kendig
Hughes STX

Abstract

A problem-solving environment called WISE uses object oriented software design to link research models from various subdisciplines for more comprehensive ecosystem simulation and visualization. A simulation manager component uses a clock to control the coupling mechanism, allowing dynamic querying between encapsulated models for the most recent values of state variables. Case studies are given which describe the potential use of this tool for ecological research.

Introduction

As human activities pervasively alter the physical systems that support life on earth,[1] we face the difficult scientific problem of predicting how terrestrial ecosystems respond to global change. The most reliable information we have is for near-term responses measurable by short-term experiments. For example, pores in the leaves of plants in forest canopies close or open as concentrations of carbon dioxide in the local air rise or fall. Leaves may experience daily changes in the level of external CO_2 that are as great as changes expected in the free atmosphere from the burning of fossil fuels over the next few decades. We can measure the rapid response of leaves to daily change; however, this near-term response may not tell us much about the long-term effects of chronically high CO_2 levels, nor about the reliability of ecological simulations designed to predict long-term response.[2]

To extend our knowledge beyond experimentally tractable short-term physics and physiology to the long-term responses of whole terrestrial ecosystems, researchers construct process-oriented simulation models. Some tested computer applications of limited scope and predictive ability are already in regular use for ecological forecasting and environmental management. For many purposes, though, we are still at an earlier stage, using computational systems to research *how* to model ecology. Good research models can demonstrate the effects of ìwhat ifî assumptions and hypotheses and help frame further questions, which will lead to better models and eventually to tested applications for routine use.

What makes good ecological modeling so difficult, and what makes the building of computational systems to further this goal such a challenge as well, is that *ecosystems are immensely complex.* They include millions of entitiesóliving species, chemical compounds, forms and fluxes of energyóand innumerable interactions among these factors. Ecological modeling always becomes an exercise in simplification and approximation. What must we include? What can we leave out?

Exploratory Simulation Using Coupled Models

Different scientific disciplines studying ecosystems emphasize different structures, details, and theories in their models. When approaching questions that cross disciplines, it is seldom clear at the outset whether detail central to the approach of one discipline (such as soil physics) is important to dynamic responses of the ecosystem as studied by another discipline (such as plant physiology). However, including all the latest theories and potentially important details in a single model would make that model intractable for analysis, and the comprehensive input data required would rarely even be available. Moreover, if a complex ecological model simultaneously embodied untested theoretical innovations from several disciplines, assessing the validity of any one of them would be difficult.

General Strategies for Coupling Models

There is no single accepted approach to coupling complex models. Some have taken the stance that already complex models are likely to become intractable when coupled to each other. Nonetheless, there are viable tactics for selectively coupling models to study or include particular effects, as shown in Figure 1.

One strategy (case 1, Figure 1) is to use one model, say Model B, as a source of data to estimate the parameters of some function to be embedded in Model A, a model that is to be changed or improved. Here the role of the model is similar to ecological field data that need to be approximated as a function of other variables. Similarly, in a modeling environment, Model B could provide simulated states or boundary conditions dynamically to Model A (case 2). This resembles providing Model A with a time series of observations. In both cases 1 and 2, Model B may be able to provide information for a broader range of circumstances than available observations can.

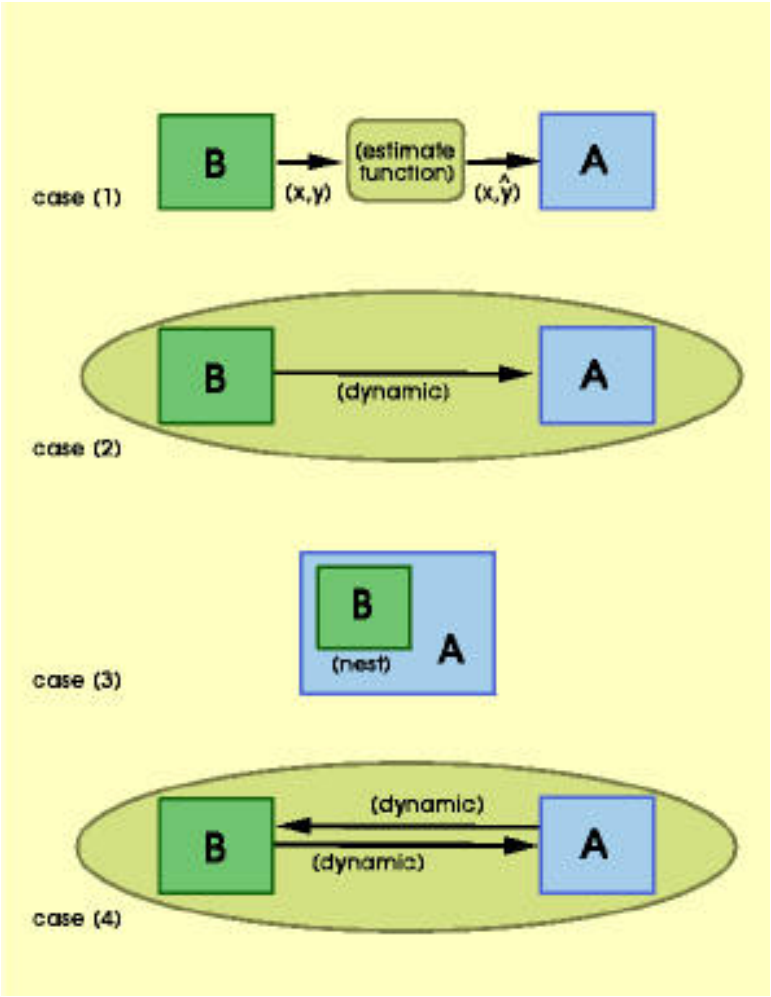


Figure 1. Four methods of coupling the behavior of one model (A) to information simulated with another (B).

In ecosystem modeling, once detail about a subsystem has proved important for answering a question, the usual approach has been to embed some version of Model B into Model A (case 3). This approach is valuable if the models are theoretically compatible, if the models have appropriately nesting representations of time and space, and if the resulting enhanced Model A remains tractable for analysis and use.

Two-way dynamic coupling of separate models can address the situation (case 4) when detail that is important to one scientific discipline affects the answers to questions asked by another. With the proper simulation environment, sufficiently short time steps, and attention to error propagation and the dynamical consequences of the coupling, models which may or may not be theoretically, temporally, or spatially compatible can be coupled to answer specific questions.

The WISE Problem-solving Environment

To help overcome the obstacles of linking models and performing coupled simulations, we have developed a problem-solving called the Workbench for Interactive Simulation of Ecosystems (WISE). WISE uses the approach described in Case 4 (Figure 1), because of its power and flexibility in dealing with complex ecological questions.

WISE lets researchers selectively test the effect of supplying to one disciplineís model the complex behavior believed to be important by experts in another discipline. Visualization tools and on-demand logging of results to files let the researcher both see the possible effect and perform further analysis of the results.

If testing with the coupled models shows that a particular simulated effect is important, then representing it on a routine basis becomes a research problem for the modelís authorójust as if experimental work or new theoretical developments within her discipline suggested the change. WISE adds to the dialogue between disciplines by providing an avenue for transferring insights among them. This approach is part of an ongoing project, an outgrowth of research into combining models from different disciplines and of applying concepts from object-oriented software design to the reuse of ecological simulation models.[3,4]

Developing System Requirements

When determining what we wanted WISE to be able to do and how, we had to consider the application domain, several issues of overall architecture, and the userís need for effective visualization.

Domain Requirements

WISE is designed to simultaneously run a variety of ecological models, or multiple instances of a particular model, for a specified simulation period. We achieved this by creating a coupling mechanism that allows any model instance to query another model instance dynamically for the most recent values of its state variables. For example, a soil model could be linked to a model of canopy growth with minimal effort, and then the canopy model could use the reported soil conditions to improve its calculations.

Because WISE is a research tool, we designed it for flexibility. For instance, models using various time steps need to run simultaneously: an hourly model might need to run in tandem with a daily model. Moreover, we wanted to be sure that scientists could test their hypotheses by modifying the internals of the model. With such a feature, a compiled ìmicroversionî of the original could then be executed using the existing interface.

A user interface that gives a common look and feel across disparate models was critical to hiding the inherent complexity of a multimodel application, especially since each model type has its own unique set-up parameters, constraints, and input files. Shared visualization tools were also needed to provide dynamic displays for monitoring selected attributes during a simulation run.

A simple encapsulation process for running a legacy model within the application was needed. By encapsulating a legacy model within an object oriented model class, we can hide implementation details and enforce common behavior between different model types. The class also allows access to the models' state variables, which are needed for visualization and intermodel exchanges. At the same time, the encapsulation process must be light-weight and minimally intrusive to the model's native code. This reduces the effort to encapsulate a model and helps minimize the chances of introducing new software errors. Furthermore, as researchers release new versions and revisions of their models throughout most development projects, updates must be easily incorporated, or the application will fall behind the state of the art in ecology.

WISE also allows for multiple levels of human expertise and interaction, including canned configurations to help novice users learn how to use the models and the PSE, while retaining flexibility for the expert.

Architecture Requirements

The first issue in designing the systemís architecture was careful management of computer resources. Often simulation modeling is very CPU-intensive, especially when many model instances are running simultaneously. Input and output files can be very large and require lots of disk space. Therefore, the system had to efficiently scale up to large simulations.

Complex configurations of models might run for hours or days on a single researcherís problem, implying a need for a multi-user environment. We addressed this need by creating separate workspaces for each researcher and maintaining them with proper user authentication.

Interprocess communication is essential since the models must be able to exchange data. In WISE, actions to be taken by an object are requested by event-driven communications, and synchronized message passing is used to exchange deterministic results between processes.

Finally, we wanted WISE to be easily portable. We enhanced portability by isolating language-dependent extensions and system-level access, and by avoiding proprietary software packages.

Visualization Requirements

We wanted WISE to provide both real-time feedback with the ability to track the evolution of a quantity in time, and a facility for postprocessing of numerical results from the simulation. Given the massive amounts of data generated by the encapsulated models, that meant we needed both visual displays and utilities for generating data files.

WISE includes a time-filtering option to compare results from models running at different time steps. For example, when running an hourly model and a daily model simultaneously, a researcher may wish to compare the average daily value of attributes from the hourly model with the daily model. In some cases, she may want to compare several configurations of a model type or the effect different models have on a given attribute. Multimodel reports allow easy comparison of attributes from different instances. In addition, some specialized ìsmartî graphics reporters get the information they need directly from particular models without burdening the simulation manager with this task.

Software Design of WISE

An object oriented approach was chosen to impliment the design in order to best achieve the goal of minimal encapsulation effort when a new model is added or an existing model is updated. When a new model is encapsulated into WISE, the class structures enforce uniform behavior between model types and allow for maximum code reuse between models. Each encapsulated model inherits its behavior from a base class, which forms a common interface between all model types and WISE. Therefore, as the WISE design evolves, modifications need to be made only to the base class, and the model is modified only to the extent of encapsulation into the base class. A set of tools for each model type was needed to assist in configuring input data to a model run. Rational Rose[6] was used to design and engineer an OO solution to obtain maximal reuse and functionality. Therefore, we are able to minimize the encapsulation process and reduce intrusions into the native model.

To distribute heavy processing, graphical display, and disk storage to the most appropriate computer, we chose a client-server architecture. Collaborating computers pass set-up files and results using remote procedure calls. Figure 2 displays an overview of the major components of the design and their interactions.

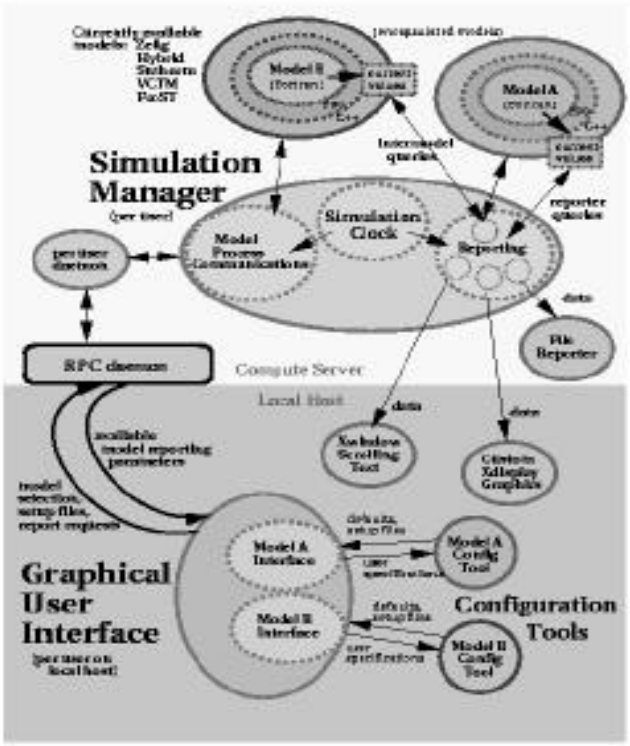


Figure 2. Context diagram of WISE, displaying major client and server entities. The arrows indicate directional information flows.

Model processes are needed to report state information to other coupled models or for visualization. We achieved this using message queues for interprocess communication on the same host. An IPC message queue class was written to hide message-passing complexities.
The Standard Template Library[7] is used in WISE as an important tool for storing and managing various objects and object types, without adding significant development costs.

Graphical User Interface

The domain requirements for ease of use lead naturally to a graphical user interface. The client-server implementation allows the models to execute on a powerful computing server while maintaining the fast response of a local GUI. Since the system supports a wide variety of models, the GUI design must be able to extract the issues shared by all models and present a common interface to the researcher.

The basic tasks to do for a simulation are
(1) select the model(s),
(2) set up or configure the model(s),
(3) set up reports for desired model results, and
(4) interact with the simulation itself.

The tasks are the same for all models, although the options available for configuration and reports are model-specific. We addressed this by a tiered design that distinguishes between simulation-level and model-level options. The main window, which remains visible throughout the simulation, provides an overview of the system options, system status, and access to simulation control (run, suspend, resume, reset, and halt). The Motif-style menu bar offers File, View, Models, Control, and Help options. Selecting a model invokes a similar window, the model main window, with options of File, View, Configure, and Help, as shown in Figure 3. Then, these common top-level menu choices cascade to model-specific options.

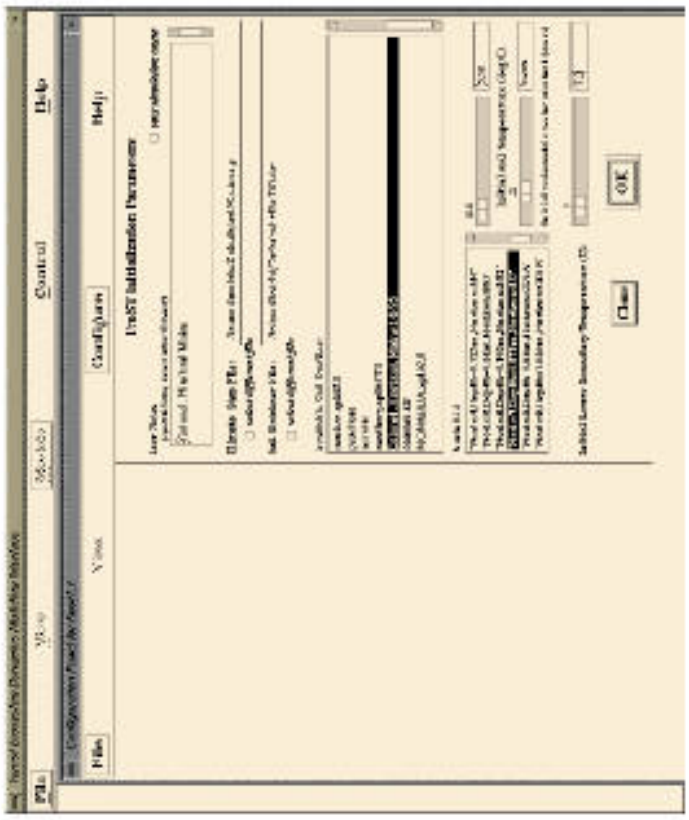


Figure 3. A sample model configuration panel with soil data selected from a list of soil profiles and initial conditions set for one soil layer.

Simulation-level Menus. A new simulation begins from the File menu. This is a fresh start with no models selected, and the user first enters certain simulation-level information via a pop-up window. Alternatively, the Open option restores a previously saved state of configured models and reports. This feature is very powerful: we can save and restore a complex array of models and reports with a single user operation. This is valuable not only for saving time but also for reliable duplication of results and for documentation of special multimodel system configurations. Save and Save As options are available at any time in the simulation to save the current workspace information. This menu also provides an Exit option from the simulation environment.

The researcher sets up multimodel reports through the View menu, which displays a list of configured models. Selecting a model invokes that modelís specific attribute selection list, as well as the report selection options detailed later.

The Models menu lists all the available types of models and provides access to microversions via a remote file selection box, which includes files on the computing server rather than the local machine. When a model or microversion has been selected, a model main window appears. One ìmodelî that *must* be selected is the Framework. Its role is special: it provides the simulation-wide settings that define the context for the rest of the models. The

Framework requires a valid user identifier on the computing server, which permits user authentication and file ownership for the simulation.

The Control menu includes the Run, Suspend, Resume, Reset, and Halt options, which are passed to the computing server to control the simulation. Reports may be added during a suspended state.

Model-level Menus. The File menu options include the steps needed to prepare a model for simulation. The GUI helps the researcher sequence through the steps properly by controlling the sensitivity of the menu options. For a new model, the first choice is the model configuration. The Standard option yields a safe setting of model parameters; the Open option allows the user to select an alternate set of parameters (saved from a previous work session). In either case, the user can display and change the associated parameter values from the configuration menu. Choosing Add Model adds an instance of this model to the GUIís internal list, and initiates the steps needed to run it. The first step is to validate the parameter settings against model-specific constraints. If the configuration is flagged as invalid, the GUI will still keep track of the model but not complete the operation. If all is well, the GUI will create a model initialization file in a model-specific format and pass it to the simulation manager. Save and Save As options allow the configuration and reports information to be saved to a disk file for later use.

Adding a model instance to the simulation (that is, creating and initializing a simulation process for the model and configuration) enables the View menu, where the researcher may specify runtime reports such as disk files and stripcharts. Selecting a reporter type triggers a sequential display of dialog boxes that cue for reporting time interval, attribute selection, and time-filtering options. These windows are generic, but the attribute list and level of reporting vary among models. For instance, a soil model may have simulation-level attributes and attributes for discrete soil layers. A dialog box will prompt for any additional information needed.

Every model has configurable items, but a simple model may only have a few grouped into one initialization window, while a more complex model may have many items organized into several categories. In both cases the Configuration menu item cascades to a list of that modelís configuration windows. Selection of a configuration menu item displays a window showing the current settings of the parameters included under that heading. These items may then be altered interactively, as shown in Figure 3.

GUI Implementation. The GUI development itself was simplified by the use of TAE+, an interface builder originally funded by NASA.[8] This tool is built on top of Motif, and the runtime libraries are freely distributable, allowing the application code to run on any Motif-compliant platform. TAE+ allows interactive widget layout, which is perfect for rapid prototyping of the windows before doing any coding. This permits a fast validation cycle with the model author for model-specific configuration windows.

Minimal coding effort is required to add a new model to the GUI by the use of shared event handlers. While the number of configurable items varies among models, the type of items does not. A relatively small set of widgets handles the existing models, and every instance of a given widget uses the same event handler.

The most complex task associated with running a model is configuring it properly. Since the GUI supports diverse models, we decided to hide most model-specific information from it by developing a configuration tool as a back end to the GUI. The model-specific configuration tool behaves as a database, with the added responsibilities of constraint checking and creation of a model initialization file in the format required by the specific model type. The constraints consist of rules and valid data ranges for initialization parameters. The configuration class accesses these constraints from an external table as a set of rules about the parameters for a model, allowing revision of constraints without having to rebuild the tool set. A reference to the set of constraints to enforce is part of each configuration. Each configuration tool is a separate process that exists for the duration of the work session, and uses a unique message queue for communication on a per-model-instance basis. This scheme makes it easy to keep track of many models and multiple configurations per model type, and greatly simplifies the GUI design.

One goal of the GUI is to make the complex model configuration process accessible to people who are not completely familiar with the model. The organization of the configuration windows aids this by organizing the parameters in a logical manner (for example, all parameters dealing with a particular subsystem would be grouped in

a separate window). Each model has a default, or safe, configuration, which may be changed interactively through the configuration windows. Simultaneously, the GUI maintains a dialogue with the configuration tool, apprising it of the current values, and receiving notice of any constraint violations. If someone enters a value for a parameter that is known to be outside its physical limits, a warning will appear in a separate window, allowing the user to undo or edit the offending item.

Managing the Computations

An RPC daemon handles incoming client calls from multiple users. A second user-specific daemon process maintains distinct workspaces for each user. This process leads to subsequent processes that are required to conduct the simulation, as shown in Figure 2.

Simulation Manager. This process controls all model and reporting instances. Initially, it creates a work area for each model instance and instantiates each model and reporter request. The simulation manager includes three classes of reporters for reporting specific state information for a particular model instance at specified intervals. File reporters write attribute values out to a file for possible postprocessing. Display reporters use visualization tools to display their values. Intermodel reporters link models. A model instance may query an intermodel reporter to obtain data from another model. Intermodel reporters are unique in that they are not initiated through the GUI but by a configured model instance.

The manager contains a clock object that accesses a list of all configured models and reporters. When a run command is issued, the clock negotiates the optimal clock step size based on the various step sizes of the configured models and reporters. The clock broadcasts a step and step size to each model and reporter object. Each of these objects determines if it is appropriate for it to act on this particular clock step. Then the clock checks the message queues for any requests from models for sharing attribute values between models. After receiving acknowledgments from all the models, the clock advances to the next step.

The researcher may suspend the simulation at the end of any clock step to add reporters or examine model output in detail. If a display is no longer needed, the user can delete the stripchart and its reporter. Creating and deleting of stripcharts on the fly can be very useful as a diagnostic tool. There is also provision via the GUI to introduce a delay between clock steps to pace output for easier comprehension.

Encapsulated models. Generally coding in Fortran, authors encapsulate their models into WISE by the following steps. The model is modified to become a re-entrant subroutine which may be called at each time step. All loops with time-related indices are replaced with conditional blocks based on the clock step. Procedural calls then are added to update attribute values in the states database. These calls are coded in Fortran 90 to allow for polymorphism based on attribute type. The model is compiled with a Fortran 90 compiler and linked to a C++ layer, which supplies the model class constructs. Models in other languages could use the same model class but would require new interlanguage calls. The model class enforces common behavior between all models and contains the repository of model attributes that are available for external queries from reporters. Finally, a configuration tool for the GUI must be created. The amount of effort required to encapsulate the model varies widely, depending on the complexity of the model and its initial inputs, but typically these steps can be achieved with a few days of effort.

WISE allows the researcher the flexibility to create variants of the default encapsulated model. These microversions are different in that the researcher provides an absolute path to them at initialization. They may also differ from the default model in the attributes available, the algorithms used for particular ecosystem processes, or in coupling to other models via intermodel queries.

Visualization Tools

Each model manages its own list of attributes, or reportable state variables. A user can request this list from the GUI dynamically at runtime, when each attribute is available for reporting in several forms. Generating a reporter request formalizes the selection of one or several attributes for feedback. Each request is sent to the simulation manager, which creates a reporter to gather the information from the appropriate models at runtime. The GUI provides a sequence of pop-up windows to help the researcher request a reporter with the desired reporting options, including attribute selection and time filters. The selected attributes may be written to a disk file for postprocessing offline, or

reported live graphically as the simulation runs. We concentrated development efforts on the stripchart, which permits simultaneous display of multiple attributes on a common scale, and scrolls over time. However, the object-oriented design of the graphics software permits adding more visual display types with little effort.

The stripchart can be used by all models, but more sophisticated graphics displays may be available, depending on the model. For instance, the Dynamic Layers graphics program reads model state variables from a many-layered soil physics model called Frost. The variables are used to generate a time series display of the soil layers, as shown in Figure 4. This graphics program is generically useful for models that employ a similar spatial structure, but it is not applicable to all models.

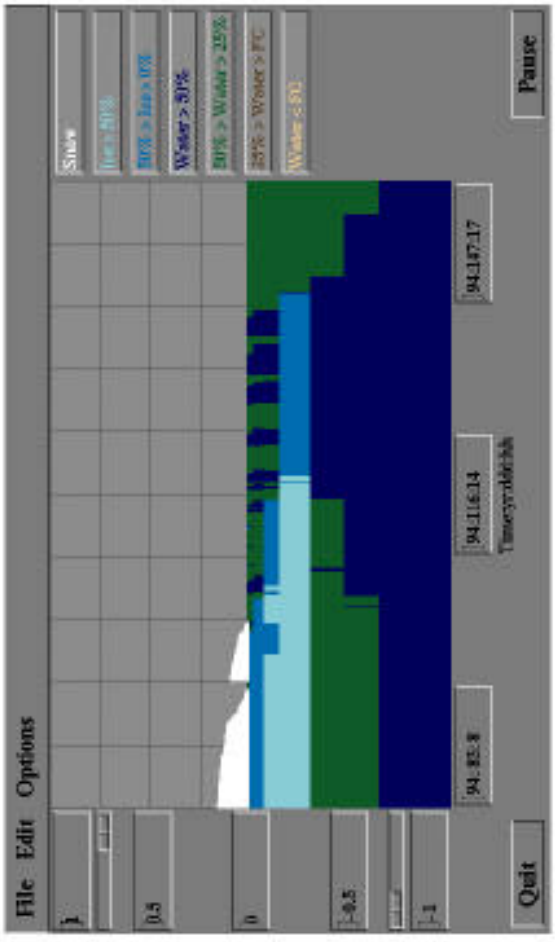


Figure 4. Snow, water, and ice content for simulated soil layers during the spring thaw of 1994. This Dynamic Layers display, more sophisticated than the all-purpose WISE ìstripchartî visualization tool, scrolls as time progresses.

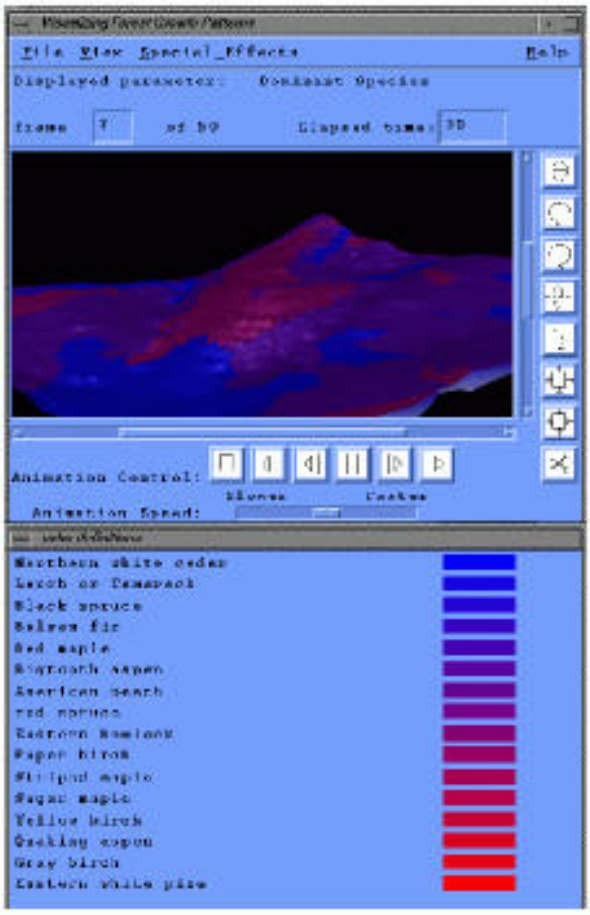
The reports are sent to the simulation manager as they are requested. This allows the user to customize the look of the display prior to runtime. Options for both the stripchart and the Dynamic Layers display include adjusting scaling in the data axis (vertical), selecting color for all components of the window, and defining line attributes for the stripchart (color, width, and style). While these may also be changed during the simulationóa pause button allows for a momentary cessation of data handlingóit is convenient to set things up beforehand as much as possible.

Finally, this implementation will permit the inclusion of 3D graphics using OpenGL by taking advantage of the ability to use OpenGL calls to drive a drawing area widget under Motif.

Post-processing

The stripchart option within WISE provides immediate feedback on selected variables as the simulation progresses, but is not sufficient to give an overall picture of the model behavior. By using the WISE "file report" option for output, however, the required state variables are printed to a file at designated time steps as the model runs. The final output file can then be used by custom post-processing software which supports statistics or mathematical routines, GIS, or other types of analysis and visualization programs.

For example, Zelig, a gap model of forest growth, was run for a simulation period of 250 years to predict species composition based on soil type and water saturation level for a site in Howland, Maine. The site contains several soil polygons , which are composed of a number of soil seres (e.g., Westbury, Colonel). The model was parameterized and runusing WISE for each soil series across the Howland site. Output data was generated for species composition, average biomass, leaf area, stem density, and depth to water table, and printed to a file. A GIS system was used to obtain a triangulation of the Howland topography and boundaries for the soil polygons. Post-processing software ingested both the GIS data and modeling output files, as well as mixing parameters for the soil series within a polygon. Derived variables were then mapped to the soil polygons using a weighting algorithm based on the composition of theseries . Figure XXXgives an example of the 3-d display derived from the combination of Zelig model output using WISE for each soil series, and GIS.



Using this method, the WISE environment, although not spatially explicit itself, can provide the means to view the model results in a way that is more amenable to comparison with field data and is easily interpreted by the researcher.

Figure XXX. Example of 3-d display derived from the combination of Zelig model output using WISE for each soil series, and GIS. Data from the Zelig model was captured at steps throughout the simulation. The software supports animation of the steps so that one may see how the patterns change with time. At any point, the viewing parameters may be interactively changed by zoom or rotation. A clipping option permits one to `chip' away at the 3-D data set to easily see the relationship of the water table and the soil surface. Tree species are colored according to their water tolerance: the most tolerant, Northern white cedar, is pure blue, least is Eastern white pine which is displayed as pure red, and the remaining species are ranked between these in varying proportions of red and blue according to a measure of their water tolerance.

The PSE At Work: 2 Case Studies

Case Study 1. Linking Models Of Different Disciplines

Using WISE, we showed that the dynamics of soil ice in a complex soil model can be important to the timing and amount of transpiration of water by forest trees simulated for cold boreal regions (such as much of Alaska, Canada, and Siberia). For example, simulations for a site in the US state of Maine, near the southern boreal forest boundary, show marked effects of soil ice in some years and not others. We simulated this with the Frost soil physics model, and a vegetation model, Hybrid, in coupled and uncoupled modes.

The Frost simulation model estimates several values of interest to soil scientists, such as water content, matric potential, temperature, and ice content within each soil horizon.[9] Frost uses network analysis to describe heat and moisture transfer, and accounts for short-wave and long-wave radiative transfer, changes in energy status, surface runoff, infiltration, redistribution, drainage, evaporation, transpiration, Penman demand, and snow. Daily input requirements include global short-wave radiation, maximum and minimum air temperatures, average wind speed, precipitation, and lower boundary soil temperature and water. Soil profile, site, and canopy characteristics are also needed. Frost is unique in that it simulates soil moisture and temperature flux at a greater level of detail, and canopy processes at a more simplified level of detail, than most ecosystem models.

Hybrid is a numerical biogeochemical model of terrestrial ecosystem dynamics[10] with features including fully coupled carbon, water, and nutrient cycles within the soilñplantñatmosphere system; drivers and constraints for climate, nitrogen deposition, and atmospheric CO_2 and O_2; and both transient and equilibrium responses to climate change.

Hybrid uses a flexible subdaily time step for cycling and exchange of carbon, nitrogen, and water, and for growth of grass. Individual tree growth occurs on an annual time step. Individual trees and grass compete for light, water, and nitrogen within a patch or plot, resulting in smaller plants being shaded by larger ones, and larger plants taking up more of the available water and nitrogen. The soil is represented as three layers with a simple daily hydrological budget. Decomposition is calculated using an empirical submodel.

In the first months of 1994, early winter snows were light in east-central Maine and soils froze to a substantial depth. Simulated latent heat fluxes from instances of Hybrid running alone and coupled to Frost were similar for much of the winter of 1994. However, during the spring thaw, the coupled model showed evaporation being restricted by soil ice (see Figure 6, in which evaporation is proportional to the latent heat flux shown). In contrast, the Hybrid-only run showed more soil moisture being lost to respiration during the thaw.

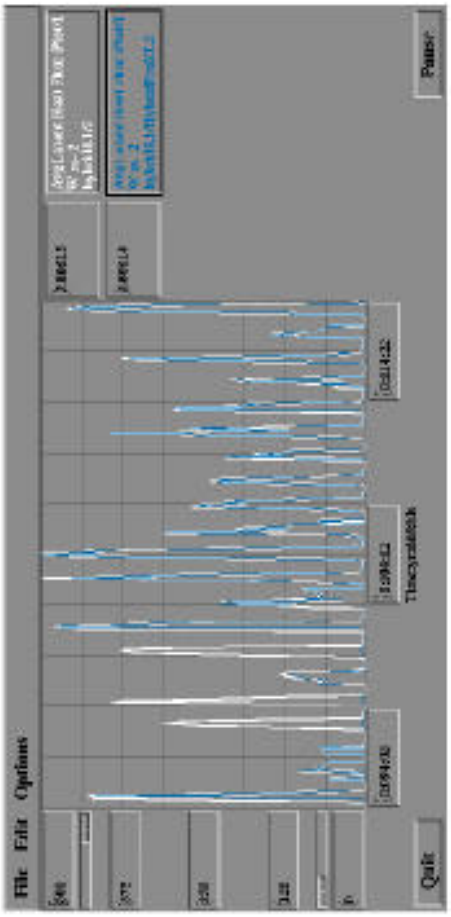


Figure 6. Simulation results for coupled ecosystem models show the usefulness of the WISE problem-solving environment. Latent heat flux (associated with evaporation) predicted by the vegetation model Hybrid alone (*white line*) mostly agrees with predictions of Hybrid coupled with the soil-physics model Frost (*blue line*). However, the results diverge markedly during the spring thaw, with important consequences for the water and carbon balances the rest of the year.

A consequence of this discrepancy is that months later in the simulation, Hybrid running alone predicted drought in August and September 1994, whereas the coupled model (with its carry-over of winter precipitation in the form of soil ice) showed sufficient soil moisture for plant growth during the whole summer season. The stand-alone Hybrid simulation lost carbon (a measure of vegetation productivity), while the coupled model had a positive carbon balance for 1994 despite below-average rainfall. Because of these findings and similar results for other boreal ecosystems, we are exploring ways of including the effects of soil ice in global simulations with Hybrid.

Case Study 2. Deriving Improved Input Parameters For Standalone Models

Linking models from different disciplines is one way of assuring that the complexity required to describe ecosystem conditions is adequately represented. Having 2 models running simultaneously, however, may be difficult in situations where resources are limited, and simulation time is important. Another way of addressing the problem of adequately parameterizing a model of one discipline with the proper inputs of another is to link different models

intially, derive output parameters from the results of the linked models, and then use critical output components as inputs to a standalone model of a single discipline. An example of how WISE was used to accomplish this follows.

The dynamics of snow in northern forest ecosystems are important for accurately assessing the energy budget during initial snow fall, over the winter, and during snow melt. The amount and duration of ice as well as available soil water are dependent upon the characteristics of snow fall at a given site. The FroST model simulates snow in a very simple way by modifying characteristics of the surface soil node and any snow covered canopy and litter nodes to represent the characteristics of a snow pack (i.e. short wave reflectivity and transmissivity, long wave transmissivity, emissivity, bulk density, and thermal conductivity). Precipitation events increase the node thickness, and a simple melt factor removes the snow. As the snow melts, node characteristics revert to that of the original soil layer. Input values for the variables above are required to accurately represent snow characteristics for a given site. Values can be obtained from literature research, from actual field measurements, or from output from a more complex and rigorous snow model.

SNTHERM.89 [11], is a one-dimensional mass and energy balancing model for predicting temperature and water profiles through snow. Snow and soil layers are subdivided into horizontally infinite control volumes which are subject to governing equations for heat and mass balance. Water flow is simulated using a gravity flow algorithm, with phase-changes, water flow, and temperature coupled by a freezing curve. A range of winter conditions are covered within the model including snow depth, rainfall, freeze-thaw cycles and transitions between snow cover and bare soil. SNTHERM.89 also predicts snow accumulation and ablation, as well as snow density and metamorphosis and their result on optical and thermal processes. Despite its complexity in simulating snow conditions, the input soil charactersitics and algorithms describing soil processes in SNTHERM.89 are much more simplisitic, and less flexible for actual field conditions than are used in FroST.

For this case study, the goal of using WISE was to derive appropriate snow input parameters for specific climate and site conditions in Saskatchewan and Manitoba, Canada that could be used to run the FroST model in standalone mode. In the first step, WISE was used to run FroST in linked mode with SNTHERM.89. FroSt provided SNTHERM.89 with information about soil moisture and temperature changes for specific soil profile characteristics over the course of the simulation. SNTHERM.89 simulated snow conditions that affected the water and heat fluxes in FroST. Since the basic time step for SNTHERM.89 is determined by the time step of the meteorological data input, it was set to run on an hourly time step as is used in the FroST model. Input climate data requirements for SNTHERM.89 were the same as that for FroST with the addition of approximate average barometric pressure over the period of the simulation, near IR extinction coefficient for the top snow node, snow albedo, and irreducible water saturation for snow. Output files of SNTHERM.89 predictions of snow density and thermal conductivity over the simulation were generated from which mean values were determined. The mean snow density and thermal conductivity information from the more complex snow model were then used as input parameters for FroST's simpler snow algorithms to allow FroST to run by itself.

Comparisons of snow depth were made between the FroST model results, with input parameters generated from the linked simulation with SNTHERM.89, and data measured in 1994 at sites in Saskatchewan and Manitoba, Canada that were collected as part of the Boreal Ecosystem Atmosphere Study (BOREAS) [12]. By using the linked models predicted parameters, simulated snow depths agreed well with observed snow measurements for 1994 (Saskatchewan $r^2 = 0.86$, Manitoba $r^2 = 0.92$).

These case studies demonstrate some of many effects that can be explored and characterized with WISE that would be difficult to study with less flexible approaches to coupling complex ecological models. By allowing these sorts of modeling experiments, WISE can contribute to better understanding of ecosystem dynamics and help lead to more informed choices about the management of ecosystems and global change. The WISE problem-solving environment could also be adapted for research on other types of complex, heterogeneous systems in other disciplines.

The six-year-old WISE development project is made up of a small team of scientists and engineers that has varied between two and five people. WISE runs on Sun architectures, specifically Berkeley and System V. Client software is also compatible with HP-Unix, and porting to other Unix platforms should be possible with modest effort. The

models vary in size and complexity but are typically a few thousand lines of code each. More information on WISE and each of the models can be found at our Web site at http://fedwww.gsfc.nasa.gov.

Acknowledgments

Jon Ranson and Darrel Williams developed the multidisciplinary research effort in forest ecosystem dynamics that provided the impetus for this work. Among others who contributed significant time, advice, or ecological models were William Bidlake, Uday Bindingnavle, Steve Fifer, Andrew Friend, Kevin Hammond, Rachel Jordan, Lara Prihodko, Hank Shugart, Jim Smith, Dean Urban, and John Weishampel. This research was supported by NASAís Office of Mission to Planet Earth.

References

1. P.M. Vitousek, ìBeyond Global WarmingóEcology and Global Change,î *Ecology*, Vol. 75, No. 7, Oct. 1994, pp. 1861ñ1876.
2. E.B. Rastetter, "Validating Models of Ecosystem Response to Global Change,î *BioScience*, Vol. 46, No. 3, Mar. 1996, pp. 190ñ198.
3. E.R. Levine et al., ìForest Ecosystem Dynamics: Linking Forest Succession, Soil Process and Radiation Models,î *Ecological Modelling*, Vol. 65, Nos. 3ñ4, Feb. 1993, pp. 199ñ219.
4. R.G. Knox et al., ìA Framework for Integrating Environmental Models to Simulate Forest Ecosystem Dynamics,î *Proc. Third Intíl Conf./Workshop on Integrating GIS and Environmental Modeling*, CD-ROM, Natíl Center for Geographic Information and Analysis, Univ. of California, Santa Barbara, 1996. Also at http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/program.html.
5. G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd ed., Benjamin/Cummings Publishing, Redwood City, Calif., 1994.
6. Rational Rose, ver. 3.0.2, Rational Software Company, 2800 San Tomas Expressway, Santa Clara, CA 95051`
7. D.R. Musser and A. Saini, *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*, Addison-Wesley, Reading, Mass., 1996.
8. M.R. Szczur and S.B. Sheppard, ìTAE Plus: Transportable Applications Environment Plus: a User Interface Development Environment,î *ACM Trans. Information Systems*, Vol. 11, No. 1, Jan. 1993, pp. 76ñ101.
9. E.R. Levine and R.G. Knox, ìModeling Soil Temperature and Snow Dynamics in Northern Forests,î *J. Geophysical Research, Atmospheres*, 1997, pp. 29,407-29,416
10. A.D. Friend et al., ìA Process-Based, Biogeochemical, Terrestrial Biosphere Model of Ecosystem Dynamics (Hybrid v3.0),î *Ecological Modelling*, Vol. 95, 1997, pp. 249ñ287.
11. Jordan, R., *A one-dimensional temperature model for a snow cover*. CRREL Special Report91-16., 49 pp., U.S. Army Corps of Engineers, Cold Regions Research andEngineering Lab., Hanover, New Hampshire, 1991.
12. Sellers, P., F. Hall., H. Margolis, B. Kelly, D. Baldocchi, G. den Hartog, J. Cihlar, M. G. Ryan, B. Goodison, P. Crill, K.J. Ranson, D. Lettenmaier, and D.E. Wickland." The Boreal Ecosystem Atmosphere Study (BOREAS): An Overview and Early Results for the 1994 Field Year", Bull. Am. Meteo. Soc., 76, 1995, pp. 1549-1577.
13. Flerchinger, G.N. and K.E. Saxton.. "Simultaneous Heat and Water Model of a Freezing Snow-Residue-Soil System I. Theory and Development". Trans. ASAE., 1989. pp. 565-571.

Robert G. Knox joined the Biospheric Sciences Branch of NASA Goddard Space Flight Center as a research scientist in 1991. His responsibilities combine research in forest community and ecosystem ecology with design of models, statistical analyses, and support of a satellite mission to study global forests. His research emphasizes the study of ecosystem constraints on vegetation dynamics, and feedbacks among vegetation dynamics, soil processes, and forest structural properties. Knox has a PhD in botany from the University of North Carolina (1987) and an AB in biology from Princeton University (1978). E-mail, knox@spruce.gsfc.nasa.gov.

Virginia L. Kalb is a software developer at the Terrestrial Information Systems Branch of NASA Goddard Space Flight Center. On the WISE project, she has primary responsibility for graphical user interface development,

contributes to the design of the modeling environment, and has written the custom interactive graphics software. She is exploring new ways to visualize the model dynamics for both model testing and for insight into the multivariate data space of ecological modeling. Kalb has a BA and MA in mathematics from the University of Maryland at College Park and is currently working on her PhD in applied mathematics. E-mail, gk@cheshire.gsfc.nasa.gov.

Elissa R. Levine has been a soil scientist in the Biospheric Sciences Branch of NASA Goddard Space Flight Center since 1986, where she has primary responsibility for designing, testing, and implementing the soil models used in WISE. She is also a principal investigator on the Global Learning and Observation to Benefit the Environment education project. Levine received her BS in natural resource management from Kansas State University and her MS and PhD (1984) from Pennsylvania State University in agronomy (soil genesis and morphology). E-mail, elissa@lichen.gsfc.nasa.gov.

David J. Kendig joined the WISE project in 1995 as a principal programmer/analyst employed by Hughes STX. He performs ongoing systems analysis and system design of computer programs for the multidiscipline, interactive modeling of forest ecosystems. Previously, he was operations manager of the Egret instrument aboard the Compton Gamma Ray Observatory. Kendig has an MS in computer science from Johns Hopkins (1996) and a BS in astronomy from Pennsylvania State University (1983). E-mail, djk@forest.gsfc.nasa.gov.

Corresponding author: Virginia L. Kalb, Code 922, NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA.