An Open Geographic Modeling Environment

by
Thomas Maxwell
(maxwell@kabir.cbl.cees.edu)
and
Robert Costanza
(costza@cbl.cees.edu)

University of Maryland
Institute for Ecological Economics
Box 38, Solomons, Md. 20688

ABSTRACT

Developing the complex computer models that are necessary for effectively managing human affairs through the next century requires new infrastructure supporting high performance collaborative modeling. In this paper we describe our Open Georgaphic Modeling Environment, which supports 1) modular, hierarchical model construction and archiving/linking of simulation modules, 2) graphical, icon-based model construction, 3) transparent distributed computing, and 4) integrating multiple space-time representations. This environment, which transparently links icon-based modeling environments with advanced computing resources, allows users to develop models in a user-friendly, graphical environment, requiring very little knowledge of computers or computer programming. Automatic code generators construct spatial simulations and enable distributed processing over a network of parallel and serial computers, allowing transparent access to state-of-the-art computing facilities. The modeling environment imposes the constraints of modularity and hierarchy in program design, and supports archiving of reusable modules in our Modular Modeling Language (MML). An associated library of "module wrappers" will facilitate the incorporation of legacy simulation models into the environment. It is hoped that this type of infrastructure will open the simulation arena to a much wider set of participants, and facilitate the application of computer modeling to the study of complex multi-scale processes in support of policy making on many levels.

Keywords: Landscape Modeling, Distributed Modular Simulation, Simulation Environment, Spatial Modeling.

1. Introduction

Ecological systems play a fundamental role in supporting life on Earth at all hierarchical scales. They form the life-support system without which economic activity would not be possible. There are many signs that the collective global economic activity is dramatically altering the self-repairing aspects of the global ecosystem. Our ability to change economic and ecological systems, and the rate of spread of the impacts of these changes, far exceeds our ability to predict the full extent of these impacts. Protecting and preserving our natural life-support systems requires the ability to understand the direct and indirect effects of human activities over long periods of time and over large areas. Computer simulations are now becoming important tools to investigate these interactions.

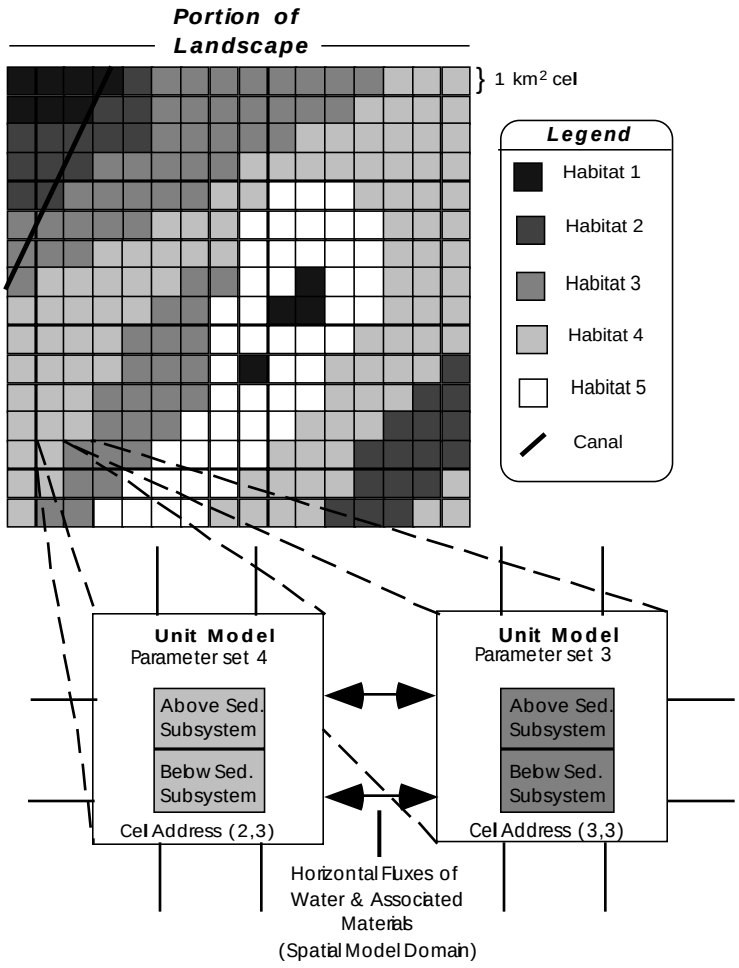

Figure 1. The basic structure of a spatial ecosystem model. Each cell has a (variable) habitat type, which is used to parameterize the unit model for that cell. The unit model simulates ecosystem dynamics for that cell in the above-sediment and below-sediment subsystems. Nutrients and suspended materials in the surface water and saturated sediment water are fluxed between cells in the domain of the spatial model.

Spatial (geographic) modeling of ecosystems is essential if one's modeling goals include developing a relatively realistic description of past behavior and predictions of the impacts of alternative management policies on future ecosystem behavior (Risser et al., 1984; Costanza et al., 1990; Sklar and Costanza, 1991). Development of these models has been limited in the past by the large amount of input data required, the difficulty of even large mainframe serial computers in dealing with large spatial arrays, and the conceptual complexity involved in writing, debugging, and calibrating very large simulation programs. These limitations have begun to erode with the increasing availability of remote sensing data and GIS systems to manipulate it, and the development of parallel computer systems. Although the importance of advances in the parallel processing field have been recognized in the field of spatial modeling (Casey and Jameson, 1988), the conceptual complexity involved in building complex models in a distributed computational environment remains a major barrier to the utilization of these systems in the environmental sciences. We propose to address this issue through the development of a spatial modeling environment to support ecological/economic (or other, e.g., biological/social/physical) model development for state-of-the-art parallel and serial computer systems. This system, which links graphical tools for developing self-contained component models with module databases and parallel code generators, will support modular, reusable model development, and allow scientists to utilize state-of-the-art parallel processing architectures without investing unnecessary time in computer programming.

2. Spatial Ecosystem Modeling

We define a dynamic spatial model as any formulation that describes the changes in a spatial pattern from time t to a new spatial pattern at time t+1, such that

$$\mathbf{X}_{t+1} = f(\mathbf{X}_t, \mathbf{Y}_t)$$

where $\mathbf{X}(t)$ is the spatial pattern at time t and $\mathbf{Y}(t)$ is a set of array, vector or scalar variables that may affect the transition. Although many forms of dynamic spatial modeling are utilized within the broad field of ecology (Sklar and Costanza, 1991), and the spatial modeling tools described here are applicable to a wide range of modeling tasks in the biological, social, economic, and physical sciences, our primary focus in this paper is on process-based landscape models. These models simulate spatial structure by first compartmentalizing the landscape into some geometric design and then describing flows within compartments and spatial processes between compartments according to location-specific algorithms (Figure 1). Examples of process-based, spatially articulate landscape models include wetland models (Sklar et al., 1985; Costanza et al., 1986; Kadlec and Hammer, 1988; Costanza et al., 1990; Boumans and Sklar, 1991; White et al., 1992), oceanic plankton models (Show, 1979), coral reef growth models (Maguire and Porter, 1977) and fire ecosystem models (Kessell, 1977).

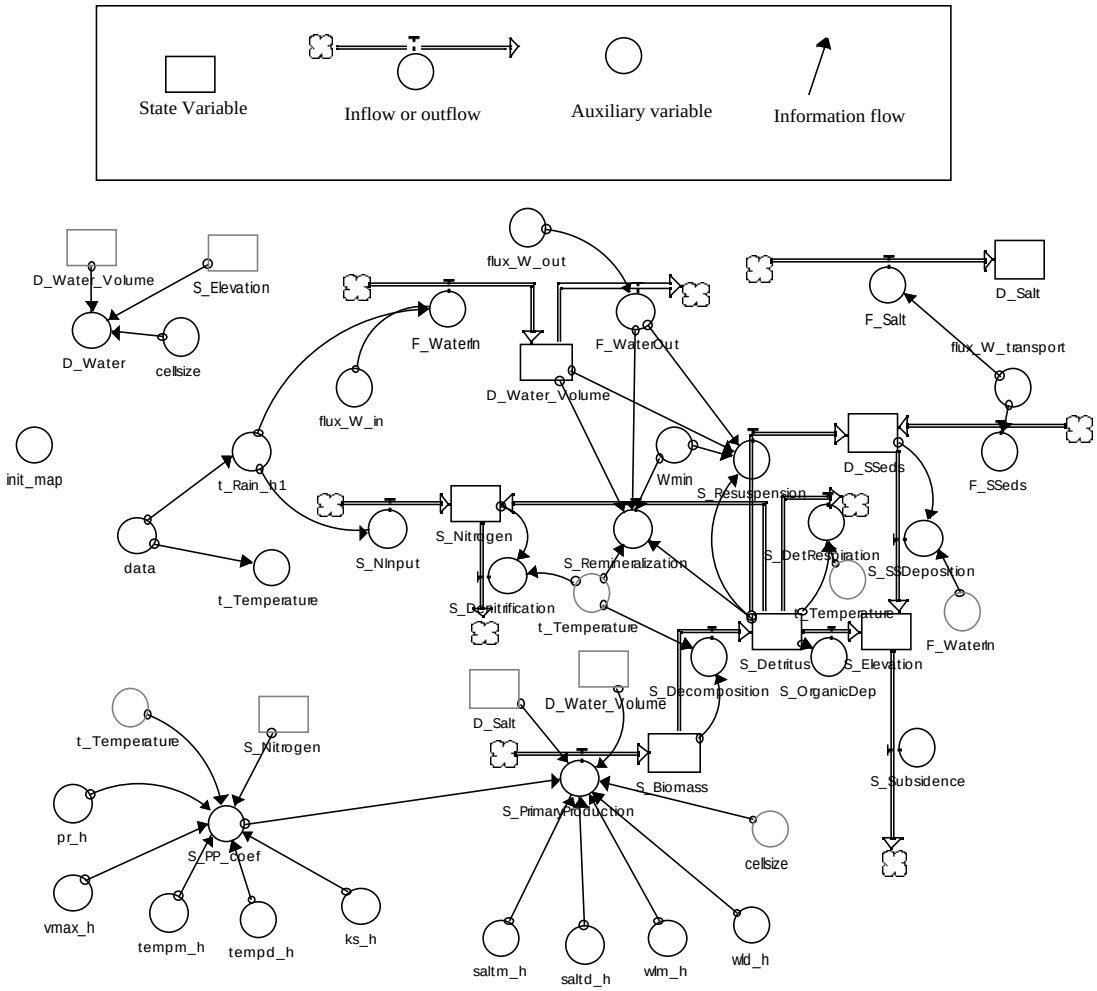

Figure 2. STELLA diagram of the unit model used for the CELSS landscape model.

One example of a process-based spatial simulation model is the Everglades Landscape Model (ELM), discussed in section 3. Another example is the Coastal Ecological Landscape Spatial Simulation (CELSS) model, which consists of 2,479 interconnected cells, each representing 1 km^2, constructed for the Atchafalaya/Terrebonne marsh/estuarine complex in south Louisiana (Sklar et al., 1985; Costanza et al., 1990). Each 1 km^2 cell in the CELSS model contains a dynamic, nonlinear ecosystem simulation model with seven state variables, similar to the one shown in Figure 2. The model is generic in structure and can represent one of six habitat types by assigning unique parameter settings. Each cell is potentially connected to each adjacent cell by the exchange of water and materials. This model is several years old and is much simpler than many models in use today.

The original CELSS model took four people about four years (16 person-years) to fully develop and implement using a supercomputer. The model has proved to be very effective at helping us understand complex ecosystem behavior and guiding policy and research (Sklar et al., 1985; Costanza et al., 1990). We are now

concerned with reducing the time involved for both developing and running this type of model, and moving the modeling to smaller, less expensive computers. Toward that end we have developed the integrated spatial modeling environment described below.

3 Everglades Landscape Model

The Everglades Landscape Model (ELM) (Fitz et al., 1993) has been developed by researchers at the University of Maryland using the Spatial Modeling Environment, Version 1 (section 6). The ELM is designed to be one of the principal tools in a systematic analysis of the varying options in managing the distribution of water and nutrients in the Everglades. Central to these objectives is the prediction of vegetation change under different scenarios. Water quantity, and the associated hydroperiod, has been a central issue in understanding changes to vegetation of the Everglades (Davis, 1994; White, 1994). Nutrients from agricultural areas also appear to be important in understanding vegetation succession (Davis, 1991) in this historically oligotrophic system (Steward and Ornes, 1975). The interaction of these factors, including the frequency and severity of fires, appears to drive the succession of the plant communities in the Everglades (Duever, 1984; Gunderson, 1989; White, 1994). Thus this system has myriad indirect interactions, constraints, and feedbacks that result in complex ecosystem structure (biotic and abiotic components and their flow pathways) and function (the modes of interaction and their rates). For this reason, it is critical to develop a systems viewpoint towards understanding the dynamics inherent in that ecosystem structure and function. Part of this process is the development of a dynamic spatial simulation model. The ELM is that analytic tool.

In this model, the important ecosystem processes that shape plant communities are simulated within the varying habitats distributed throughout the landscape. The principal dynamics within the model are: plant growth in response to available sunlight, temperature, nutrients, and water; flow of water plus dissolved nutrients in three dimensions; fire initiation and propagation; and succession in the plant community in response to the environment. Using a mass balance approach in incorporating process-based data of a reasonably high resolution within the entire Everglades landscape, changing spatial patterns and processes can be analyzed within the context of altered management strategies. Only by incorporating spatial articulation can an ecological model realistically address large scale management issues within the vast, heterogeneous system of the Everglades.

For the spatially explicit ELM, the modeled landscape is partitioned into a spatial grid of 10,178 square unit cells, each having 1 km2 surface area. The ELM is hierarchical in structure, incorporating an ecosystem-level "unit" model (Fitz et al., 1995) that is replicated in each of the unit cells representing the Everglades landscape. The unit

model itself is divided into a set of model sectors that simulate the important ecological (including physical) dynamics using a process oriented, mass balance approach. While the unit model simulates ecological processes within a unit cell, horizontal fluxes across the landscape occur within the domain of the SME. Within this spatial context, the water fluxes between cells carry dissolved nutrients, determining water quantity and quality in the landscape.

4. Conceptual Complexity and Model Development

Development of ecosystem models in general has been limited by the ability of any single team of researchers to deal with the conceptual complexity of formulating, building, calibrating, and debugging complex models. The need for collaborative model building has been recognized (Goodall, 1974; Acock and Reynolds, 1990) in the environmental sciences. Realistic ecosystem models are becoming much too complex for any single group of researchers to implement single-handed, requiring collaboration between species specialists, hydrologists, chemists, land managers, economists, ecologists, and others. The current generation of models tend to be idiosyncratic monoliths that are comprehensible only to the builders (Acock and Reynolds, 1990). Communicating the structure of the model to others can become an insurmountable obstacle to collaboration and acceptance of the model. Policy makers are unlikely to trust a model they don't understand.

A well-recognized method for reducing program complexity involves structuring the model as a set of distinct modules with well-defined interfaces (Gauthier and Ponto, 1970; Goodall, 1974; Tichenor, 1989; Acock and Reynolds, 1990; Hodges et al., 1992; Silvert, 1993). Modular, hierarchical model structuring is well developed in the context of discrete event modeling (Zeigler, 1976; Zeigler, 1990), but has received comparatively little development in the realm of continuous modeling (Goodall, 1974; Cellier, 1991; Silvert, 1993). Ecosystem models with a modular hierarchical structure should be closer to natural ecosystem structure than procedural models (Goodall, 1974; Silvert, 1993), since the component populations of ecosystems are themselves complex hierarchical systems with their own internal dynamics. Modular design facilitates collaborative model construction, since teams of specialists can work independently on different modules with minimal risk of interference. Modules can be archived in distributed libraries and serve as a set of templates to speed future development. The inheritance property of object-oriented languages allows the properties of object-modules to be utilized and modified without editing the archived object. A modeling environment that supports modularity could provide a universal modeling language to promote worldwide collaborative model construction.

A second step toward reducing model complexity involves the utilization of graphical, icon-based module interfaces, wherein the structure of the module is represented diagramatically, so that new users can recognize the major interactions at a glance. Scientists with little or no programming experience can begin building and running models almost immediately. Inherent constraints make it much easier to generate bug-free models. Built-in tools for display and analysis facilitate understanding, debugging, and calibration of the module dynamics.

One major advantage of this graphical approach to modeling is that the process of modeling can become a consensus building tool. The graphical representation of the model can serve as a blackboard for group brainstorming, allowing policy makers, scientists, and stakeholders to all be involved in the modeling process. New ideas can be tested and scenarios investigated using the model within the context of group discussion as the model grows through a collaborative process of exploration. When applied in this manner the process of creating a model may be more valuable than the finished product.

5. Computational Complexity and Parallel Processing

Tremendous computational resources are required to integrate the equations of a large spatial model in a reasonable amount of computer time. Large models typically require supercomputers for efficient execution. This class of models is a near-ideal application for parallel processing since a typical model consists of a large number of cells that can be simulated semi-independently. Each processor can be assigned a different subset of cells, and most interprocessor communication is nearest-neighbor only. Despite their great promise and increasing availability, parallel architectures have not found much usage in the life sciences. The major barrier to wide acceptance of these techniques has been the difficulty of programming and debugging large parallel programs, and reluctance on the part of scientists to invest time in learning new languages and architectures.

High performance computing must be transparent in order to be generally accessible. The user should not be concerned with the details of what platform the simulation is running on or how the simulation is being farmed out to different processors- these details are handled automatically by the environment without the users knowledge or intervention. However, supporting transparent distributed computing for spatial modeling requires infrastructure development, as described later in this paper.

6. Geographic Modeling Environment

In an attempt to address the conceptual and computational complexity barriers to dynamic geographic model development, we have developed the spatial modeling environment (SME), which links icon-based graphical modeling environments with parallel supercomputers and a generic object database (Maxwell and Costanza, 1995;

SME2, 1995). This system will allow users to create and share modular, reusable model components, and utilize advanced parallel computer architectures without having to invest unnecessary time in computer programming or learning new systems. The following sections give a brief description of the current design of the SME. A more detailed description can be found in the web page (SME2, 1995).

The SME design has arisen from the need to support collaborative model building among a large, distributed network of scientists involved in creating a global-scale ecological/economic model. Its design should eventually be general enough to support most large scale modeling tasks within the biological/environmental sciences. In the interest of maximizing accessibility to a distributed network of collaborators, the system is designed to support a range of platforms, both in the front-end development environment and in the back-end distributed network of platforms. Since our goal is to support modularity in model design and separate the implementation of the model dynamics from the details of simulation code development, it has been necessary to abstract the module libraries from both the front-end and the back-end environments. We are thus led to the formulation of a three-part Modelbase-View-Driver architecture (Figure 5). The three components are described below.

6.1 View

The View component of the SME is used to graphically construct, calibrate, and test biological/ecological modules. This component is represented by an off-the-shelf graphical modeling environment such as STELLA, Simulab, or Extend.

6.2 Modelbase

In the next step toward constructing a spatial model, the Module Constructor translates the View ecosystem component modules into Module objects defined in our text-based Modular Modeling Language (MML). The MML objects can then be archived in the ModelBase to be accessed by other researchers, and/or used immediately to construct a working spatial simulation. Many MML objects can be combined hierarchically in the MML. This MML hierarchy can then be converted by the Code Generator into a C++ object hierarchy within the spatial modeling environment (SME), where it can drive a spatial simulation.

The MML is designed to capture only the relevant dynamics of the simulation module being constructed, and leave out all implementation-specific details. For example, the features that can be represented in the MML include dynamics of growth, death, and transformation of biological/ecological entities, fluxes of water, nutrients, pollutants, etc., and the internal decision and learning processes of biological agents. The features that are not represented in the MML include the spatio-temporal implementation of the model, input and output of model data,

and the distribution of the model over a set of processors. These features are implemented by the Code Generator and the simulation drivers (next sections).

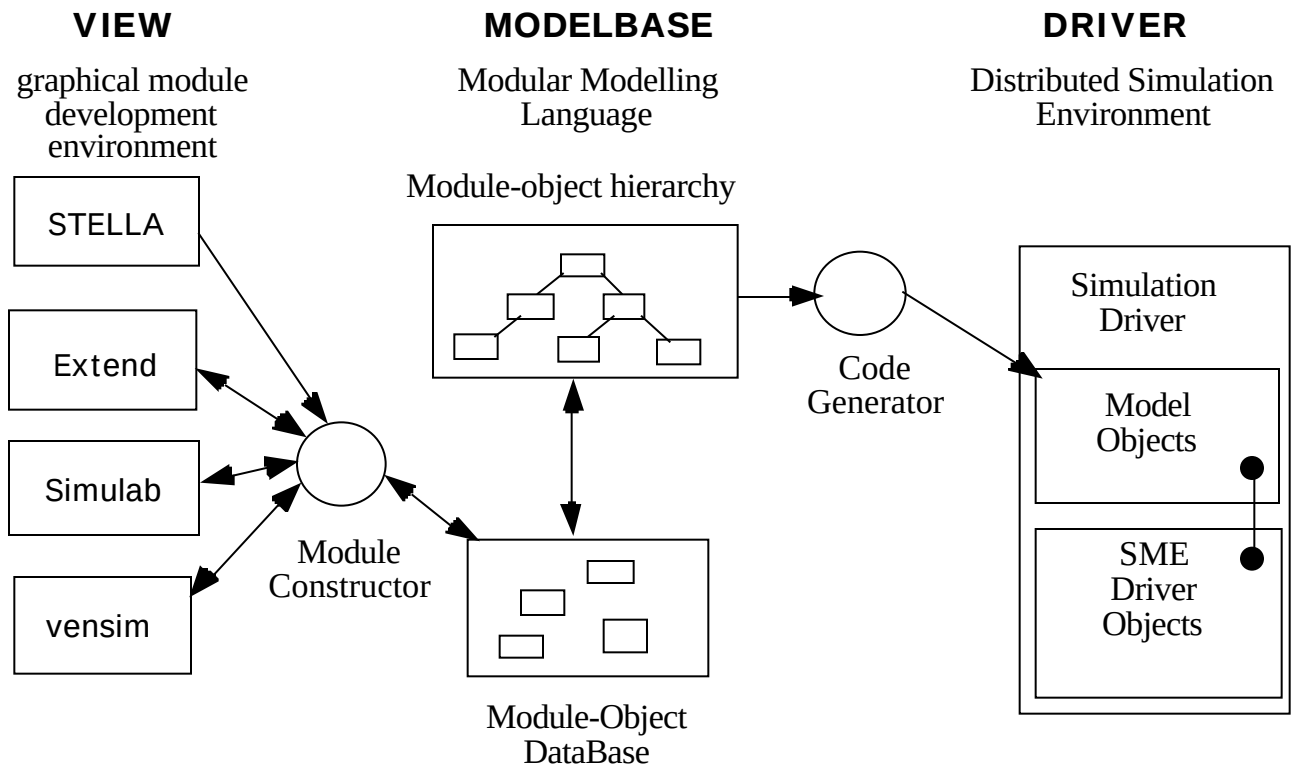

Figure 5. Overview of the Modelbase-View-Driver architecture.

6.3 Code Generators

The Code Generators convert a MML object hierarchy into a C++ object hierarchy which is incorporated into the simulation driver application to create a spatial simulation. The user customizes the set of objects generated by entering information into a set of configuration files that are initially generated by the Code Generator. In the final version a menu-driven interface will be provided to facilitate this configuration step.

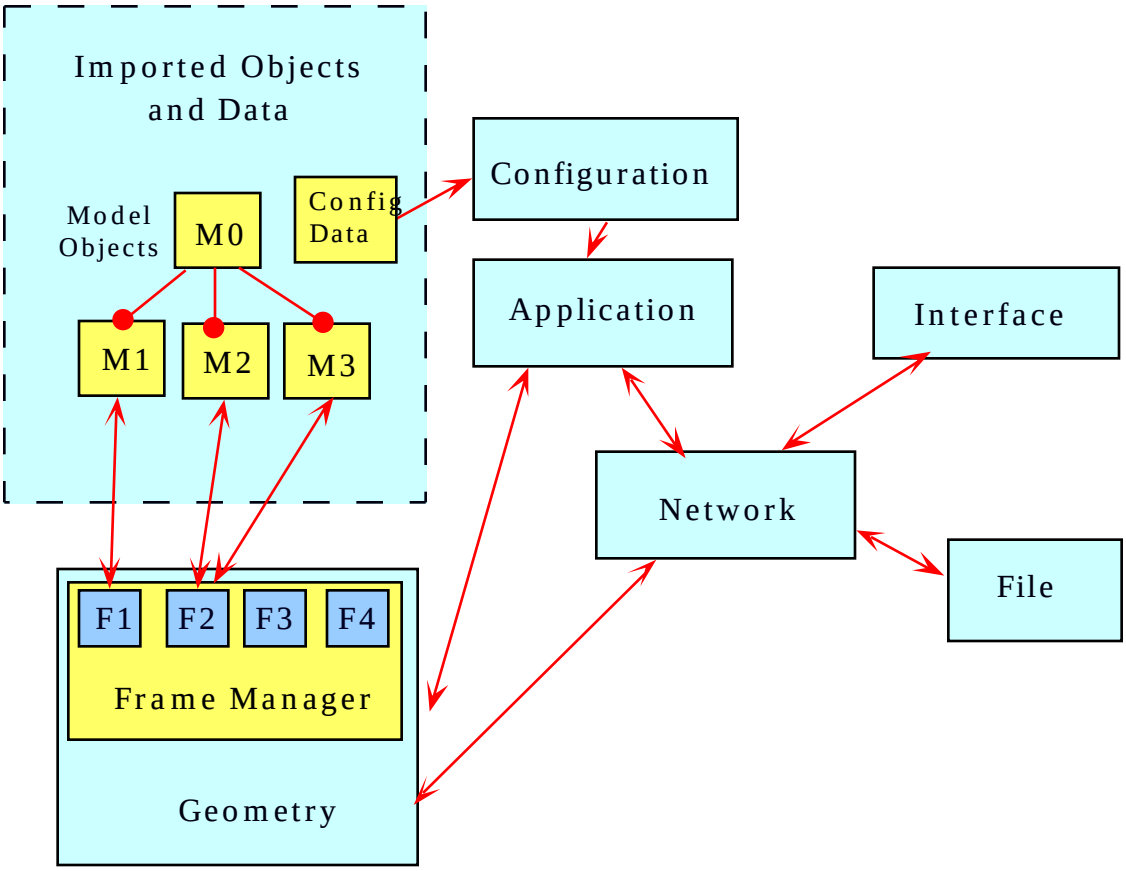

Figure 6. Structure of the SME.2 simulation driver as discussed in the web page (SME2, 1995).

During the configuration step the user specifies the additional information that is required to transform the MML object into a dynamic simulation object. The information entered falls into several general categories:

1) Space-time implementation. In this step, each MML object is associated with a frame, which specifies its space-time implementation. A frame is a C++ object which specifies the topology of the spatial implementation of the module, methods for interacting with and transferring data to other frames, and temporal methods for handling the passing of time. The driver geometry object (Figure 6) maintains a catalog of available frames. Examples of available frames include two-dimensional grids (e.g., for landscapes), graphs and networks (e.g., for river, canal, or neural networks), and agents (e.g., for individual agents moving about in the landscape). The user specifies a frame type as well as (a set of) GIS map(s) that the frame will read at runtime to configure itself.

2) Input/Output configuration. In this step the user configures input to the simulation from the biological/ecological databases and GIS. Input configuration must be done at code generation time because the Code Generator uses this information (together with the variable dependency graph) to determine variable types. Output configuration is done at runtime, although default values can be specified in the CG configuration files

6.4 PointGrid Library

The PointGrid library (PGL) is a set of C++ distributed objects designed to support computation on irregular, distributed networks and grids. It contains the core set of objects on which the SME Driver is constructed. The PGL object structure is a direct mapping of an early version of the OGIS Open Geodata Model (OGIS, 1996) to C++.

The PGL builds spatial representations from sets of Point objects (see below) with links. It transparently handles: 1) creation and decomposition (over processors) of Point Sets, 2) mapping of data over and between Point Sets, 3) Iteration over Point Sets and Point Sub-Sets, 4) data access and update at each Point, and 5) swapping of variable-sized PointSet boundary (ghost) regions. Some of the important PGL classes are:

- *Point:* Corresponds to a cell in a GIS layer.
- *Aggregated Point:* Corresponds to a cell in a coarser resolution GIS layer.
- *PointSet:* A set of Points with links (grid or network).
- *DistributedPointSet:* A PointSet distributed over processors with variable-sized boundary (ghost) layers.
- *Coverage:* Mapping from a DistributedPointSet to the set of floats.

6.5 Driver

The SME Driver (Figure 6) is a distributed object-oriented simulation environment which incorporates the set of code modules that actually perform the spatial simulation on the targeted platform. It is implemented as a set of distributed C++ objects linked by message passing. Of all the objects shown in Figure 6, only the interface object is visible to the user, the rest will perform their tasks automatically and invisibly. The major driver components include:

- Application Object. Handles general process of simulation execution and coordination and scheduling of the other SME objects.
- Imported Objects and Data: This is the set of objects and data that is created by the Code Generator and imported into the driver. The objects are C++ implementations of the ModelBase modules. The imported object structure is described in more detail below.
- Geometry Object: Maintains the catalog of frames (see section 7.3) and handles all tasks relating to the spatial configuration of the simulation, such as translating/ transferring data between frames and the network object.
- Network Object: Handles communication between processors and the simulation host. It is implemented using the MPI message passing interface standard (MPI, 1995).
- Interface Object: Menu-driver interface facilitating user control of the simulation and real-time display of simulation output. Provides the user with a single familiar environment in which to interact with simulations running on any one of a number of parallel or serial computers.
- File Object: Handles archiving of simulation output in HDF format. Later versions will actively participate in an open GIS environment using the Open GeoData formalism (OGIS, 1996).

The imported objects are built upon the following classes:

- Module class: The CodeGenerator Application converts each module in the MML model description into a Module Object in the Driver. Each Module Object has a set of Variable Objects and a Frame Object. It also has a set of methods for responding to simulation events such as initialize and update.
- Frame class: A Frame Object is a driver object which specifies the topology of the spatial implementation of a Module Object, including methods for interacting with and transferring data to other frames. A frame has a list of Point Objects (POs), with each PO corresponding to a cell in the frame's map region, which includes a partition of the study area handled by the current processor plus a communication buffer zone. The driver maintains a catalog of available frames, which includes two-dimensional grids (e.g., for landscapes), graphs and networks (e.g., for river, canal, or neural networks), areas (e.g. for embedded lumped-parameter models) and point collections (e.g., for individual agents moving about in the landscape).
- Variable class: The CodeGenerator Application converts each variable declared in the MML model description into a Variable Object in the Driver. The Variable class is a specialization of the Coverage class, which encapsulates a mapping from the set of Point Objects owned by the Module's Frame Object into the set of floating point numbers.

6.6 User Interface.

The SME user interface, which is currently under development using the tcl/tk scripting language (Tcl/Tk, 1996), will provide the user with a single familiar environment in which to build and run simulations on any one of a number of parallel/distributed or serial computers. This user-friendly environment, with hierarchically structured interaction levels, will allow users with widely varying goals and background knowledge (from scientists and students to policy makers) to build, configure, and run spatial simulations, and generate graphical output in a manner appropriate to their level of expertise.

The lowest level of interface to the SME is expedited by a tcl shell. In the SME/tcl shell environment the user can create new projects, customize the SME, and run the various SME sub-applications. Three menu-driven tcl/tk applications are being developed for 1) configuring the Driver and code generators, 2) controlling the simulation, and visualizing the simulation output, and 3) building models in MML.

- **Simulation Configuration Interface:** All simulation IO is accomplished using "pipe" objects, e.g. the real-time screen display of a variable's spatial data is rendered by configuring a pipe object to connect the spatial variable with a map animation object. The user interface provides a menu driven tool for configuring pipes for 1) map input from GIS, 2) parameter input from relational databases, 3) map output to GIS, 4) assorted data input/output to/from disk archives, and 5) various types of real-time display, including map animations, graphs, and tables. This interface also allows the user to configure various parameters associated with each simulation object.

- **Simulation Object Browser & Viewer:** A separate tcl/tk application allows the user to browse through the objects in a paused simulation and view each object's internal data structures in a convenient format. The browser also provides a menu of each object's dependent objects, so users can quickly traverse the dependency tree while search for anomalies in the simulation output. This application incorporates the viewers that are used for real-time display of simulation output.
- **Icon-Based Model Development Interface:** A third interface component is under development to provide a icon-based interface to the Modelbase component of the SME, to facilitate simulation module development and linking/archiving in the Modular Modeling Language (MML). This interface provides a graphical mapping of each component of the MML language, allowing modelers to create MML modules in a user-friendly, visual environment. The environment enforces proper MML syntax and model design, provides a blackboard for collaborative model development, and also provides on-line help screens to document each component of the MML language.

6.7 Linking Existing Simulation Code with the SME

In order to create a new module in the SME one must develop it in the View graphical modeling environment or the Modular Modeling language (MML). There is a wealth of complex simulation code in existence in the world today, written mainly in FORTRAN or C, that would be too difficult to completely rewrite in the MML or a View-supported language (although this might be the optimal approach, manpower permitting). Therefore we are developing a stand-alone version of the network object displayed in Figure 6 that will form the core of a "SME wrapper." This "wrapper" is a library of FORTRAN or C functions that a simulation developer can embed in existing "legacy" code to give it the ability to interact and exchange data with the SMP over the Internet. Once the wrapper is incorporated into the legacy simulation code, then SME variables can be linked with legacy variables using simple configuration commands. The SME and the legacy code can be run simultaneously and feed information back and forth across the Internet. For example, a SME landscape simulation might wish to link with an existing hydrodynamics simulation to handle the hydrodynamics of the watershed.

7. Simulation Development in the SME

Realistic spatial models are extremely complex, requiring large quantities of data, so that designing, calibrating, and validating these models is a difficult task. The development of a spatial simulation occurs in three stages: 1) non-spatial module development, 2) non-spatial model development (linking modules), and 3) spatial model development.

7.1 Simulation Design

The simulation design process occurs primarily in the View component, where the simulation unit modules are created. Thus the vast majority of the design work occurs in the non-spatial regime, involving concepts and

processes that are familiar to most modelers. Typically a group of modelers will work together on a set of closely related modules, with prior agreement on the set of available outputs from each module. Once developed and tested in the View environment, the modules are then linked in the View or ModelBase environment for a further round of tests. Implementing the spatial interactions can occur by designing these interaction in the MML language, or by linking predefined methods from the SME Driver libraries. Libraries have been developed to implement common hydrologic scenarios, including movement of water and constituents over and under the landscape surface. The spatial dynamics are tested in the SME Driver environment.

7.2 Calibration and Verification

The simulation calibration and verification process involves three phases:

- **Phase 1:** The unit modules are calibrated individually and non-spatially in the View environment.
- **Phase 2:** The assembled unit model is calibrated and verified non-spatially in either the View or the Driver environment.
- **Phase 3:** The full unit model is calibrated and verified spatially in the Driver environment. The final stage of calibration involves only the spatial aspects of the simulation, all calibration that can be accomplished without reference to the spatial nature of the system is completed in phase 1 and 2.

Due to data limitation, most calibration and verification in phase 3 is accomplished using sets of "integrators", i.e. localized quantities whose dynamics depend on a number of spatial processes. For example, river flux rate and nutrient concentrations may be measured at a number of stations along a river. This data plays a major role in calibrating and verifying a number of spatial processes that influence the movement of water and nutrients across the landscape and into the river.

8. Conclusions

Parallel computer hardware and software are now well developed enough to allow their use in large-scale biological and environmental modeling. Parallel systems are particularly well suited to spatial modeling, allowing relatively complex unit models to be executed over a relatively high resolution spatial array at reasonable cost and speed. When linked with icon-based graphical model development tools and GIS/database tools, one has a powerful yet easy-to-use spatial modeling environment.

In addition, the widespread use of object-based modeling environments linked transparently to state-of-the-art distributed computing resources could result in a fundamental paradigm shift in complex systems modeling. The modeling formalism imposes the constraints of modularity and hierarchy in program design. General adoption of this paradigm will support the development of libraries of modules representing reusable model components that are

globally available to model builders, as well as making advanced computing architectures available to users with little computer knowledge.

We believe that effectively managing human affairs through the next century will require extremely complex and reliable computer models. Widespread utilization of modeling environments supporting graphical, hierarchical/modular design may be essential in facilitating reliable, economical model construction.

9. References

Acock, B. and Reynolds, J.F., 1990. Model Structure and Data Base Development. In: R.K. Dixon, R.S. Meldahl, G.A. Ruark and W.G. Warren (Editors), Process Modeling of Forest Growth Responses to Environmental Stress. Timber Press, Portland, OR.

Boumans, R.M.J. and Sklar, F.H., 1991. A Polygon-Based Spatial Model for Simulating Landscape Change. Landscape Ecology, 4:83-97.

Casey, R.M. and Jameson, D.A., 1988. Parallel and Vector Processing in Landscape Dynamics. Applied Mathematics and Computation, 27:3-22.

Cellier, F.E., 1991. Continuous System Modeling. Springer-Verlag, New York, NY.

Costanza, R., Sklar, F.H. and Day, J.W., 1986. Modeling Spatial and Temporal Succession in the Atchafalaya/Terrebonne Marsh/Estuarine Complex in South Louisiana. In: D.A. Wolfe (Editors), Estuarine Variability. Academic Press, New York.

Costanza, R., Sklar, F.H. and White, M.L., 1990. Modeling Coastal Landscape Dynamics. BioScience, 40:91-107.

Davis, S.M., 1991. Growth, Decomposition and Nutrient Retention of Cladium Jamaicense Crantz and Typha Domingensis Pers. in the Florida Everglades. Aquatic Botany, 40:203-224.

Davis, S.M., 1994. Phosphorus inputs and vegetation sensitivity in the Everglades. In: S.M. Davis and J.C. Ogden (Editors), Everglades: the Ecosystem and Its Restoration. St. Lucie Press, Delray Beach, FL.

Duever, M.J., 1984. Environmental Factors Controlling Plant Communities of the Big Cypress Swamp. In: P.J. Gleason (Editors), Environments of South Florida: Present and Past. Miami Geological Society,

Fitz, H.C., Costanza, R. and Reyes, E., 1993. The Everglades Landscape Model (ELM). South Florida Water Management District, Everglades Research Division,

Fitz, H.C., DeBellevue, E., Costanza, R., Boumans, R., Maxwell, T. and Wainger, L., 1995. Development of a General Ecosystem Model (GEM) for a range of scales and ecosystems. Ecological Modelling, 88: 263-297.

Gauthier, R.L. and Ponto, S.D., 1970. Designing Systems Programs. Prentice-Hall, Englewood Cliffs, NJ.

Goodall, D.W., 1974. The Hierarchical Approach to Model Building. Center for Agricultural Publishing and Documentation, Wageningen.

Gunderson, L.H., 1989. Historical Hydropatterns in Wetland Communities of Everglades National Park. Freshwater Wetlands and Wildlife, 61:1099-1111.

Hodges, T., Johnson, S.L. and Johnson, B.S., 1992. A Modular Structure for Crop Simulation Models. Agronomy Journal, 84:911-915.

Kadlec, R.H. and Hammer, D.E., 1988. Modeling Nutrient Behavior in Wetlands. Ecological Modelling, 40:37-66.

Kessell, S.R., 1977. Gradient Modeling: A New Approach to Fire Modeling and Resource Management. In: C.A.S. Hall and J.W.J. Day (Editors), Ecosystem Modeling in Theory and Practice. Wiley-Interscience, New York, NY.

Maguire, L.A. and Porter, J.W., 1977. A Spatial Model of Growth and Competition Strategies in Coral Communities. Ecological Modeling, 3:249-271.

Maxwell, T. and Costanza, R., 1995. Distributed Modular Spatial Ecosystem Modelling. International Journal of Computer Simulation: Special Issue on Advanced Simulation Methodologies, 5(3):247-262.

MPI, 1995. Message Passing Interface. URL: http://www.mcs.anl.gov /mpi/index.html.

OGIS, 1996. The OpenGIS Guide. URL: http://ogis.org/guide/guide1.htm.

Risser, P.G., Karr, J.R. and Forman, R.T.T., 1984. Landscape Ecology: Directions and Approaches. Illinois Natural History Survey, Champaign, IL.,

Show, I.T., Jr., 1979. Plankton Community and Physical Environment Simulation for the Gulf of Mexico Region. Society for Computer Simulation,

Silvert, W., 1993. Object-Oriented Ecosystem Modeling. Ecological Modeling, 68:91-118.

Sklar, F.H. and Costanza, R., 1991. The Development of Dynamic Spatial Models for Landscape Ecology. In: M.G. Turner and R. Gardner (Editors), Quantitative Methods in Landscape Ecology. Springer-Verlag, New York, NY.

Sklar, F.H., Costanza, R. and Day, J.W.J., 1985. Dynamic Spatial Simulation Modeling of Coastal Wetland Habitat Succession. Ecological Modeling, 29:261-281.

SME2, 1995. Spatial Modeling Environment, version 2 alpha. URL: http://kabir.umd.edu /SMP/MVD/SME2.html.

Steward, K.K. and Ornes, W.H., 1975. The Autecology of Sawgrass in the Florida Everglades. Ecology, 56:162-171.

Tcl/Tk, 1996. Tcl/Tk Page at Sun MicroSystems. URL: http://www.sunlabs.com/research/tcl/index.html.

Tichenor, L.H., 1989. Modular Simulation of the Static Portions of the Leaf Budget. Simulation, 55:345.

White, M., Day, D., Maxwell, T., Costanza, R. and Sklar, F., 1992. Ecosystem Modeling Utilizing Desktop Parallel Computer Technology. In: R. Falconer (Editors), Hydraulic and Environmental Modeling: Estuarine and River Waters. Ashgate,

White, P.S., 1994. Synthesis: Vegetation Pattern and Process in the Everglades Ecosystem. In: S.M. Davis (Editors), Everglades: the Ecosystem and Its Restoration. St. Lucie Press, Delray Beach, FL.

Zeigler, B.P., 1976. Theory of Modeling and Simulation. Wiley, New York, N.Y.

Zeigler, B.P., 1990. Object-Oriented Simulation with Hierarchical, Modular Models. Academic Press, New York, NY.