A Multimodeling Basis for Across-Trophic-Level Ecosystem Modeling: The Florida Everglades Example

Paul A. Fishwick
Department of Computer and Information Science and Engineering
University of Florida

James G. Sanderson
National Biological Survey
Department of Wildlife Ecology and Conservation
University of Florida

Wilfried F. Wolff
Department of Biology
University of Miami

November 5, 1997

Abstract

We present a new modeling method for use in large-scale physical systems, such as the Everglades ecosystem. The current work that has been done in the ATLSS (Across-Trophic-Level System Simulation) project—which focuses on simulating key Everglades system components—relies on *code* integration. While this represents a necessary first step in analyzing the dynamics of species within the Everglades, it falls short of true *model* integration. We have constructed a methodology called object-oriented physical modeling (OOPM), which allows a comprehensive knowledge representation to be constructed for large scale systems. OOPM enforces the idea that an implementation of computer code can be accomplished in an incremental fashion by starting with a conceptual model and progressing to more detailed models. During this evolutionary procedure, a minimal amount of code is written since the emphasis is on developing the conceptual model so that it not only represents the intuitive aspects of the model, but that it is also executable. OOPM provides a kind of "blueprint" for ecologists, biologists and hydrologists to communicate and integrate models effectively.

1 Introduction

Modeling is an essential enterprise in ecology when combined with empirical "in the field" studies. Field work and modeling are being used in conjunction to evaluate proposed changes to the Florida Everglades as part of a nationally supported restoration effort. One problem with modeling is that most models are at different abstraction levels. For example, in studying a landscape such as the Everglades, some species' populations are homogeneous,

and so a continuum approach is warranted to track time-dependent changes in density, mass, and age structure. For other species, such as wading birds, panthers, deer, and other higher-trophic organisms, models capable of incorporating individual behavior are required. Three model types have been proposed: *general population*, *structured population*, and *individual based* models to model across-trophic-level interactions. Historically, these models are coded, often using different programming languages, and then executed for their singular purpose. However, the definition of "system" is one where interactions among species and with the landscape must be considered. We need to begin with the meaning of the words "model" and "code." We define *model* as an abstract—often visual—formal representation of a system. It is "formal" in the sense that the representation is provided with a level of clarity that affords execution of the representation on a computer. Examples of model types are: finite state machines, System Dynamics models (Forrester 1969; Forrester 1971; Richardson and Pugh 1981), Petri nets (Peterson 1981), Bond Graphs (Thoma 1975; Cellier 1991; Rosenberg and Karnopp 1983), and sets of ordinary or partial differential equations. A drawing that is informal might exhibit the facade of a model, but if it does not conform to strict syntactical rules, then we label it as an informal, conceptual model. We define *code* as representing a linear sequence of source lines written with a programming language such as FORTRAN, C, C++ or Pascal. Code *modules* represent packages or libraries of source lines. Both models and code are translated, but they are at two different abstraction levels. Code is translated into machine-translatable *binaries*—or object code. Models are translated into code. While there is only one level of abstraction separating the concepts of model and code, the distance traversed from model to code is significant. Models are economic in their representations of dynamics and represent powerful cognitive tools for reasoning about system behavior.

With reference to "blueprint" in the **Abstract**, an analogy will help to clarify the integration problem from a modeling perspective. The problems facing model integration are analogous to the carpenter, plumber, and electrician constructing a house. Each has separately created a working network: the carpenter has built the frame, the electrician has soldered wires and boxes for a completely wired house circuit, and the plumber has assembled the plumbing and heating. No blueprint has been used to guide each of the three, and so arguments arise and fast but inelegant fixes are made just to "make it all work." The three work in isolation in performing their functions. The problem with this scenario is founded on a lack of knowledge-sharing. Without a blueprint to serve as a design of house integration, the house cannot be constructed. To model the Everglades properly, many models must be written and assembled. We must find a way for the modelers to communicate by specifying their models using a common framework. Glue and paste methods may initially be necessary, but a knowledge representation and design framework must eventually be created. Only then, can we focus on models instead of code. This will ease the burden on the ecologist who wishes to use simulation while investigating species population growth and decay.

Many of the above concerns are also discussed by Zeigler (Zeigler 1984), whose work shares a common interest with ours regarding the importance of formal model representation over code specification. In particular, the *System Entity Structure*, supports a similar object-oriented knowledge representation with more of a focus on integrating formal representations. Our work concentrates on model types that represent large classes of models that are used in practice. We develop a methodology to achieve proper integration of models. First, we present background on the Everglades in Sec. 2 so that the reader knowledgeable in

simulation will gain some insight into the historical and contextual reasons why modeling is being done. We follow with a brief discussion of the current state of ATLSS in Sec. 3. We then proceed to a proposed model integration procedure using OOPM, described in Secs. 4 and 5. Methods described in these previous two sections are applied to an Everglades example in Sec. 6. Conclusions are summarized in Sec. 8.

2 Background and Motivation

2.1 Everglades Restoration History

In February 1996, Vice President Albert Gore and Carol M. Browner, Environmental Protection Agency Administrator, announced the Clinton administration's environmental restoration plan for the south Florida Everglades. The restoration of the greater Everglades landscape was declared a top environmental priority. During this century, the Everglades were successively divided and conquered. By the time the southern Everglades became Everglades National Park in 1949, the contextual setting of the park made clear that management decisions *outside* the park would eventually define what happened *inside* the park. Today, much of the south Florida landscape supports agriculture and large human populations whose demands for freshwater define the goals of water management in the region.

As a result of changes in the hydrology brought about by the need to supply human demands including flood control, many regional features have been changed (Davis and Ogden 1994). Natural ecological processes were altered and organisms responded. What was once uninterrupted sheet flow became a water flow fragmented by levees and impoundments. Extensive short hydroperiod marshes were degraded by overdrainage or lost altogether to development. Attenuated seasonal changes in water depth became more pronounced. Major drydowns in the sloughs, once rare, became more frequent. Because of water diversions from Lake Okeechobee and the Everglades, reduced flows into estuaries enabled seawater penetration further inland and also contributed to the demise of Florida Bay, a once productive fishery. The oligotrophic water of the historical Everglades became eutrophic in places through agricultural runoff and sparse sawgrass marshes changed to chocked cattail stands.

These freshwater alterations led not only to a change in the spatial and temporal distribution of living organisms but also to a decline in numbers of native and endemic species. Animal populations suffered dramatic declines and abnormal dieoffs. Before the turn of the century, millions of wading birds populated south Florida (Frohring, Voorhees and Kushlan 1988). Now these populations are estimated to be a mere 5% of their previous numbers (Ogden 1994).

Gunderson et al. (1995) discussed how government policy directed increased control of the hydrological resources of south Florida. The dramatic decline in wading bird populations has been blamed on changes in the hydrology, in particular the area and the length of time land is inundated by water. The U.S. Army Corps of Engineers was given responsibility for flood control and insuring that agricultural interests were protected (Halloway 1994). Now, the U.S. Army Corps of Engineers is responsible for the restoration of the same landscape that is expected to require decades of work and cost hundreds of millions of dollars. Already, lands north of Lake Okeechobee along the Kissimmee River have been allowed to return to a more natural flood regime as a result of actions taken to reduce flood control (Dahm et al.

1995).

2.2 Ecosystem Modeling

Dale and Rauscher (1994) reviewed the state of biological modeling. They discussed models that addressed the impacts of forests at four levels of biological organization: global, regional or landscape, community, and tree. They suggest the development of landscape vegetation dynamics models of functional groups as a means to integrate the theory of both landscape ecology and individual tree responses to climate change. Furthermore, they recommend (1) linking socioeconomic and ecological models; (2) interfacing forest models at different scales; (3) obtaining data on susceptibility of trees and forests to changes in climate and disturbance regimes; and (4) relating information from different scales.

Hunsaker et al. (1993) reviewed terrestrial ecosystem models and stressed the use of geographic information system (GIS) map layers. Though GIS is not time dependent the authors believed a macro language embedded in the GIS can be used to implement time dependent modeling. Languages such as Arc Macro Language (AML) are cumbersome at best and offer no special data structures beyond arrays. The authors note that "The problem of integrating models with different temporal and spatial scales is not trivial and requires special methodologies."

The development of models suggested by Dale and Rauscher (1994) and Hunsaker et al. (1993) depends upon a critical understanding of what is being modeled and the computing power required to achieve the desired level of understanding. We consider an ecosystem to be a relatively homogeneous area of vegetation that responds to governing physical processes fairly uniformly across the areal extent (Forman 1995). In contrast, a landscape consists of two or more ecosystems and responds to the governing physical processes in spatially and temporally varying ways. We are interested here in developing a methodology that facilitates modeling heterogeneous landscapes at different spatial and temporal scales and across trophic levels.

3 The ATLSS Project

The Across-Trophic-Level System Simulation (ATLSS) (ATLSS 1997) is a landscape scale ecosystem represented by conceptual models, computer programs and code from several different sources such as the US Geological Survey, University of Tennessee, University of Miami and University of Maryland. The plan for ATLSS is for it to be used to assist in the evaluation of proposed system-wide changes in the hydrology of south Florida. ATLSS will predict changes in landscape vegetation and multi-level organism responses to proposed changes in the hydroperiod brought about by restoration efforts. For example, ATLSS will predict the responses of wading birds to changes in system-wide water levels over a twenty year period. One potential outcome might be that wading bird populations increase for 10 years and then decline because of some unforeseen change that unfolds only after a decade of, say, vegetation change. Such unforeseen outcomes can only be studied using complex, detailed models. Restoration attempts that "do something" and then "correct it" based on short term outcomes might cause irreparable long term damage (Kushlan 1979).

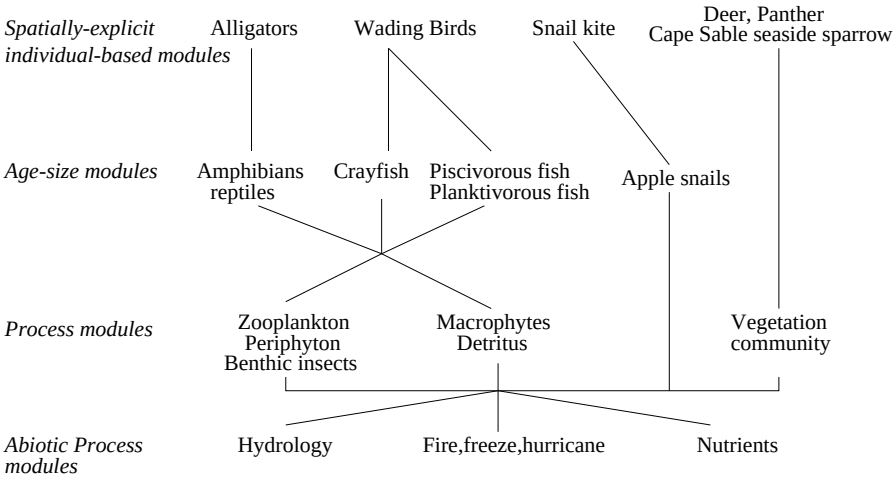

Figure 1: Planned collection of ATLSS modules (ATLSS 1997)

To simulate abiotic and biotic processes of south Florida, ATLSS was organized in a way that permitted the use of code modules that represent ecological knowledge at widely varying levels of detail and across many different ecosystems while at the same time maintaining the ability to incorporate existing legacy code. Code integration development, toward an integrated ATLSS, has been accomplished by our ATLSS partners at the University of Tennessee (UT) (ATLSS 1997), with an initial code integration "template" suggested by one of the authors (Sanderson). ATLSS requires different fidelity simulation codes to be integrated in a top down fashion. For instance, the hydrology code in ATLSS is cell based, the lower trophic organisms are coded using stable, time dependent non-linear differential equations, and the higher organisms are coded using structured population or individual-based codes (DeAngelis and Gross 1992; Wolff 1994). Because appropriate scales are used to simulate organism responses at each trophic level, reliable qualitative predictions across the south Florida landscape are possible and system-wide integrity can be better understood (Fleming, DeAngelis and Wolff 1995).

The underlying spatial structure of ATLSS is a grid of varying resolution down to $28^2 m$, that covers the study area from Lake Okeechobee in south central Florida to Florida Bay lying south of the peninsula, and east-west from coast to coast. All codes execute on some part of this grid, though not all codes are grid-based. Fig. 1 illustrates the implementation plan for ATLSS code modules.

Individual-based codes are behavioral modules that use realistic decision making capabilities of the individual organisms of the species the codes represent. These decisions must be made within the landscape occupied by the individuals. The environmental information, biotic and abiotic, available to the individual varies as a function of position. Movement and dispersal are functions of the information that is available and perceived by the organism. Animal movement and habitat selection have long been the subject of behavioral ecologists hence an organism's rules can be both complex and realistic. ATLSS represents an attempt to code the behavioral ecology of organisms at the scale of the landscape (Lima and Zollner 1996).

4 Toward Model Integration

Until the present time, the ATLSS project has evolved various code modules that have been integrated by the University of Tennessee (UT) team. A general lack of resources, and real deadlines for simulation results, has made it difficult for the UT team, and the overall ATLSS team as a whole, to research the possibility of integrating models together instead of large segments of code. At the University of Florida, our purpose is to construct a modeling language with two distinct objectives: 1) to serve as a visual modeling interface for ecologists, and 2) to serve as a potential model integration method that ATLSS might adopt in the future if enough time and resources are provided. ATLSS as it is defined and constructed today does not use models in the sense that we have defined model in Sec 1. The University of Florida initiative represents an exploratory project to determine whether ATLSS might use model integration in the future, rather than to remain an integrated system of code modules.

The first objective is consistent with the current emphasis on code module integration. The visual model is automatically translated into a code module that can be integrated directly with the remaining ATLSS code modules. All modules can then be subsequently directed to "operate" over specific time and landscape scales. The second objective is more ambitious, but we feel that the OOPM methodology represents a potentially strong candidate for a model integration method for ATLSS. We now proceed to discuss the modeling approach which satisfies the above two objectives.

5 Object-Oriented Physical Modeling

5.1 Overview

The basis for physical modeling in a large-scale system such as ATLSS begins with aggregation and object-oriented design concepts. We will briefly explore the background on both of these topics. The aggregation problem (Zeigler 1976; Zeigler 1979; Zeigler 1985) has long been a concern to simulationists and ecologists. Zeigler's DEVS formalism and the related problem of multiple aggregation levels have been applied to ecological problems whether behavioral or at the level of the landscape (Maxwell and Costanza 1995; Vasconcelos and Zeigler 1993; Vasconcelos, Zeigler and Pereira 1995; Zeigler, Vasconcelos and Pereira 1994; Zeigler and Moon 1996). Multimodeling, as originally developed by Fishwick (Fishwick 1991) in a paper on heterogeneity in high-level model integration, is a method for integrating specific high-level model types most often used by ecologists and others in science and engineering contexts (Fishwick 1995). The taxonomy of models in (Fishwick 1995) reflects the taxonomy of programming language *styles* used in computer science (Fishwick 1996a). Multimodeling is grounded in a more mathematical systems formalism such as DEVS (Fishwick and Zeigler 1992). In this sense, multimodeling (Fishwick 1995) and multiformalism (Zeigler 1976; Zeigler 1979) are co-related and at different abstraction levels. Furthermore, multiformalism and multimodeling techniques are founded on system theoretic concepts and modeling methods.

With regard to object oriented methodology, our approach has been to employ a new methodology called Object-Oriented Physical Modeling (Fishwick 1996b), which is an ex-

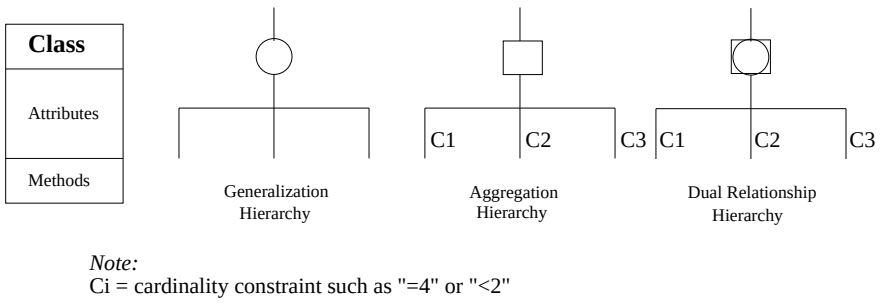

Figure 2: Structure of a class with three relations.

tension to object-oriented (OO) design as expressed by the Software Engineering OO community (Booch 1990; Rumbaugh et al. 1991). The extension is made specifically to define how physical models should be incorporated into the kind of object-oriented designs found in software engineering (Rumbaugh et al. 1991; Booch 1991; Fowler and Scott 1997). Our work is not the first to combine object-oriented design with simulation. Related work in simulation has focused on objected oriented techniques (Rothenberg 1989; Hill 1996; Zeigler 1990). Our specific contribution, relating to this work, is that we build a detailed description of how physical object dynamic and static models are mapped into an object-oriented design framework using the model taxonomy presented in (Fishwick 1995). This taxonomy divides models into categories reflective of programming language styles: declarative, functional and constraint (Fishwick 1997). Primitive models are created from these three types, and then multimodels (Fishwick 1991; Fishwick 1993; Fishwick et al. 1994; Fishwick and Zeigler 1992) are constructed to create the larger models. To create each model within a potential next-generation ATLSS, a class graph with relations among classes was constructed. Furthermore, attributes and methods within classes were identified. An object is an instance of a class. A method is an action within a class and hence within an object. For instance, there exists a base class *Bird*, and a *wading_bird* object is an instance of class *Bird*. Method *fly()* is in class *Bird*, and so *fly()* is also a method within object *wading_bird*. When we instantiate, say, 1000 wading bird objects, we are creating 1000 individual wading birds that can be modeled separately using a common set of attributes such as age, sex, and location, and methods.

The act of coding in an object-oriented language, is not ultimately a substitute for model design. As an example, C++ provides many object-oriented capabilities, but does not enforce object-oriented design. Norman (1988) suggests the need for visual, conceptual models in general design for improved user-interfaces to physical instruments and devices. The importance of design extends to all scientific modeling endeavors. Models must provide a *map* between the physical world and the virtual world produced by computers.

5.2 Conceptual Model

A key part of conceptual modeling is identifying classes. This procedure is ill-defined but some rules and approaches do exist (Fishwick 1995; Graham 1991) to aid the model engineering process. Fig. 2 illustrates generalization (◯) and aggregation (□) relations. We also permit an analyst to specify any given relation as both aggregation and generalization. This is delineated with a circle inside a square. It is not necessary to group all relations into one

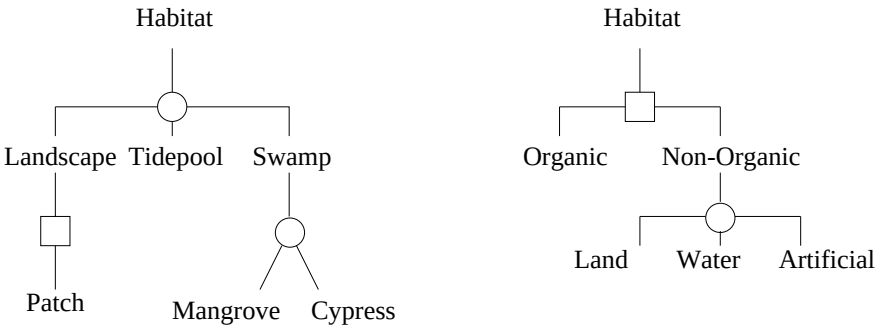

Figure 3: Conceptual Model, Part I.

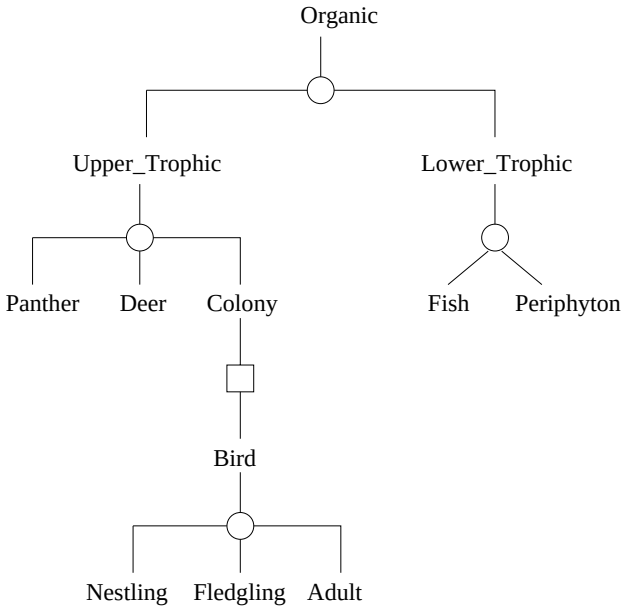

Figure 4: Conceptual Model, Part II.

graph or hierarchy—multiple graphs or hierarchies are possible. Figures 3 and 4 provide the scenario for samples classes necessary for integrated Everglades modeling. In Fig. 3, we show two trees of the conceptual model for class *Habitat*. In the left tree, *Landscape*, *Tidepool* and *Swamp* are defined as types of ecosystems or *Habitat*. Moreover, all *Habitat*-type objects are aggregates of organic and inorganic objects. In Fig. 4, there are two types of *Organic* class: *Lower_Trophic* and *Upper_Trophic*.

Our extension of object-oriented design specifies that an attribute is one of two types: *variable* or *static model*. Likewise, a method is of one of two types: *code* or *dynamic model*. A method can be of a functional (representing a function) or constraint (representing a relation) nature. Once the conceptual model has been constructed, we identify the attributes and methods for each class. An attribute is a *variable*, whose value is one of the common data types, or a *static model*. A method can be *code*, whose form depends on the programming language, or a *dynamic model*. The structure of a class is seen in Fig. 5. Variables and code are described in OO languages such as C++ (Stroustrup 1991). We define a static model as a graph of objects and a dynamic model as a graph of attributes and methods. The model

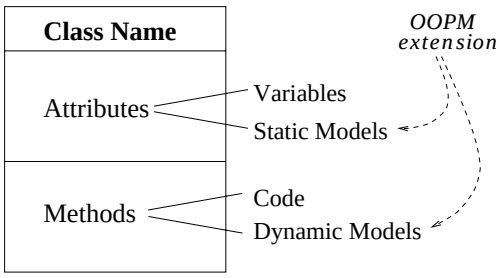

Figure 5: Structure of a Class.

types of interest here are dynamic. However, the concept of static model complements the concept of dynamic model: methods operate on attributes to effect change in an object. Dynamic models operate on static models and variable attributes to effect change.

5.3 Representing Dynamic Models

How are dynamic model components represented in the physical modeling methodology? We will illustrate two model types (functional and declarative), each with two model sub-types. For the functional model types, we use a block model and a System Dynamics model. For the declarative model type, we will use an FSA and a Petri net. The following notation, consistent with the previous notations, will be used throughout this discussion:

- *Objects:* An object is represented as obj. obj_i represents a sub-object of obj (in its aggregation hierarchy), and obj^i represents a super-object that is composed, in part, of obj. When indices i and j are used, it is possible that $i = j$ or $i \neq j$. This relation rests with the particular application.

- *Attributes:* $obj.a$ represents a variable attribute and $obj.A$ represents a static model attribute. a is short for any string beginning with a lower case letter; A is short for any string beginning with an upper case letter. Attribute references (i.e. names) and values are relevant: a name is just $obj.a$ whereas the value of attribute a is denoted as $v(obj.a)$. The following special attributes are defined for all objects: $obj.input$, $obj.output$ and $obj.state$ and represent the input, output and state of an object at the current time.

- *Methods:* $obj.m()$ represents a code method and $obj.M()$ a dynamic model method. m is short for any string beginning with a lower case letter; M is short for any string beginning with an upper case letter. The following special methods are defined for all objects: $obj.input()$ and $obj.output()$ and represent the input and output time trajectories.

The block model contains a collection of blocks coupled together in a network. Each block calculates its input, performs the method representing that block, and produces output. A block is not an object since it represents a method *within an object*. Without any refinement within a block, a function takes no time to simulate. Time is ultimately associated with state change. All obj_i represent sub-objects of obj. Fig. 6 displays a sample functional

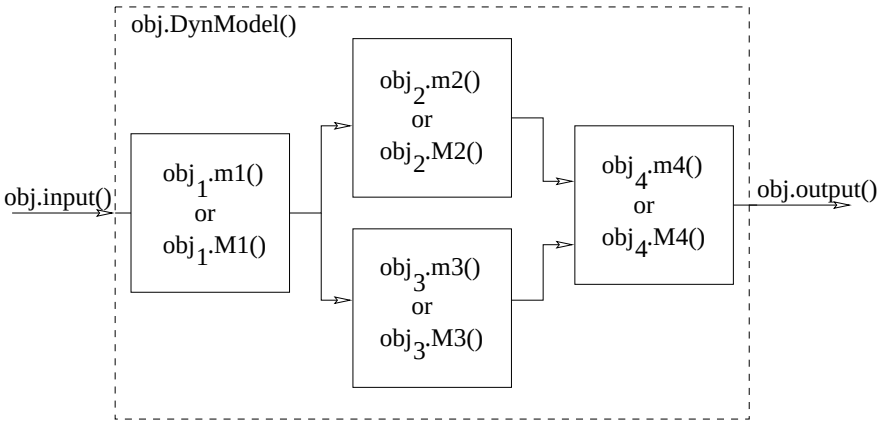

Figure 6: Functional Block model.

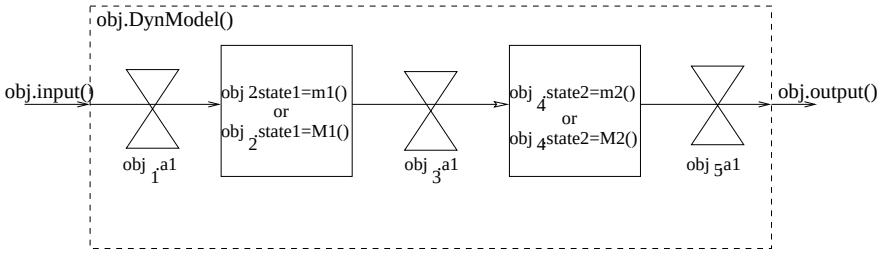

Figure 7: System Dynamics model.

model. Fig. 7 shows a System Dynamics model that is similar to the block model except that instead of methods represented by the network nodes, a node represents an object's state (a variable attribute). The same kind of multimodeling represented in Fig. 6 (refining a block into a dynamic model) can be done for the model in Fig. 7; a block is refined into a System Dynamics model. This multimodel refinement is particularly useful for our two declarative model types shown in Figs 8 and 9 where *input*() and *output*() are not as obvious unless we capture the model inside of a functional block. Multimodeling is denoted by drawing a dashed functional block around the model, denoting model refinement. The methods *input*() and *output*() are essential to perform *coupling* of models together. An FSA will have an *input*() method and a state variable attribute *obj.state*, but we require coupling information somewhere if we are to decide where the input is coming from and where the output is going to. This coupling information for a method of *obj* is present in some obj^i. For example if the FSA belongs in a bird, defining the dynamics of state change , the input must come from a physical object outside of the bird but within a more encapsulating object such as the colony.

The predicates p_1 and p_2 in the FSA model in Fig. 8 require further explanation. A predicate is defined as a logical proposition, whose value is boolean, containing internal and external events. External events are of the form *obj.input*() and internal events are of the form *obj.state* or $obj_i.state$. Rules are another convenient dynamic model (of the declarative type) that express changes of state and event. A rule-based model is composed of a collection of rules within an object *obj*, each rule i of the form: IF $p_i(event, state)$ THEN *obj.mi*() or

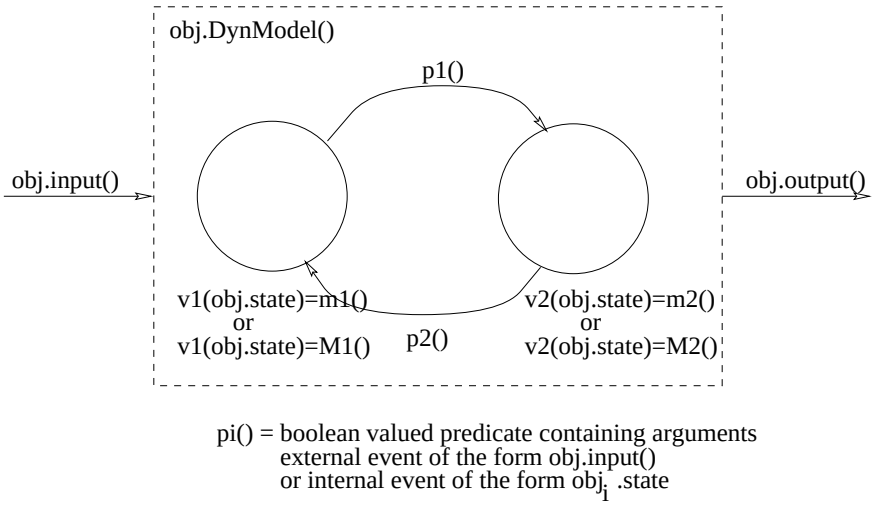

Figure 8: A finite state automaton.

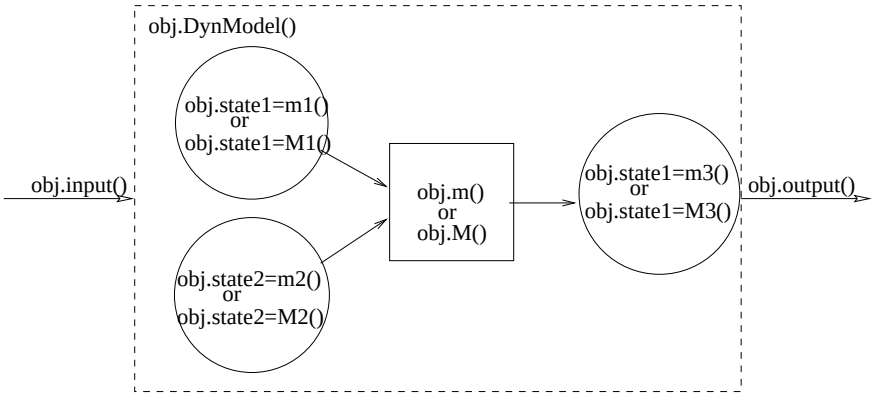

Figure 9: Petri net.

$obj.Mi()$. The phrase $p_i(event, state)$ defines the same logical proposition discussed for the FSA.

Regarding coupling and closure under coupling of model components, the rule is that coupling on a single level (i.e., for components within a model at the same level of abstraction) is performed by coupling the output of one function f_1 into the input of another f_2 to obtain $f_2(f_1())$. This operation represents functional composition. Inter-level coupling is achieved by implementing any function as one of the $M_i()$ or $m_i()$ that are specified in the above figures. The outside of every model type is a function with an input and output, and the inside has *multimodel entry points* in the form of functions M_i and m_i. For example, Fig. 8 is a function with an input and output as designated by the functional block surrounding it. However, new functions may be substituted for the m_i and M_i defined as part of the model. This makes it so that the FSA can be embedded within a larger model as one if its components, and that the FSA contain refined models of different types that are refined through the m_i and M_i functions. Currently, no provision is made for type checking at the couplings, although this would be a valuable addition in the implementation.

Multimodels are executed in the following manner. For any given scenario, execution begins by invoking the method `Execute()` within the most general composing object. Hybrid behavior involving both event scheduling and numerical integration is handled uniformly by forcing all events to be routed through a single future event list. Continuous behavior is interpreted to involve discrete events with regular time spacing.

6 Everglades Modeling

To apply the OOPM methodology to the Everglades scenario, we first construct a conceptual model using classes and the aggregation and generalization relations. Figs. 3 and 4 display hierarchies for two classes: *Habitat* and *Organic*. This provides a logical structure to the physical objects found in the Everglades. The following are sample relations drawn from the conceptual models in Figs. 3 and 4:

- *Landscape*, that is composed of multiple *Patches* is a type of *Habitat*. From this, we can create an object called *everglades* of type *Landscape*.
- *Colony* is an aggregate of *Bird* of which *Adult_Bird* is a type (i.e., class).
- *Fish* represents a type of *Lower_Trophic* form of life.

This conceptual model serves as a base to define attributes and methods. We proceed to focus mainly on dynamic model methods of importance to the Everglades. There are three prominent model types employed: 1) general population, 2) structured population, and 3) individual-based models. In the first type, one constructs a model that represents the dynamics of an entire population without regard to transitions within the population or species structure. General population models have wide coverage in the literature since they are by far the most utilized model type, mainly due to the ease of closed-form solution and simple form (Murray 1990). Unfortunately, this simplicity may be artificial rather than reflecting true behavior. For more accurate models, one can structure a population by physiological attributes such as age or size (Hallam et al. 1992; Ulanowicz and Platt 1985). The third model type encompasses models where each individual in the population is modeled separate from the others. The modeling of periphyton is a general population model, without structure, whereas fish are modeled using structure. Upper trophic animals such as panther, deer, alligators, and wading birds, such as wood storks, are modeled individually.

DeAngelis et al. (1992; 1992) create a structured hierarchy for these model types. A *p-state* model corresponds to a general population model. An *i-state* model has two subtypes: distribution and configuration. An *i-state distribution* model corresponds to the structured population model and an *i-state configuration* model corresponds to modeling each individual. Two sample papers deal with specific tradeoffs with the two *i-state* approaches (DeAngelis and Rose 1992; DeAngelis et al. 1993). Wolff (Wolff 1994) refines the individual approach into two sub-categories: individually-based models and individually-oriented models. An individually-based model (IBM) is one where individuals are modeled, but generally with equation structures that are common among all individuals. An example of this is provided by Hallam et al. (Hallam et al. 1992) for the change in biomass for an

individual as a function of time and age structure:

$$\frac{\partial}{\partial t}m_i + \frac{\partial}{\partial a}m_i = g_i(m_1, m_2, \ldots, m_n) \tag{1}$$

where m_i represents the mass of individual i and a is age. Note the homogeneity in this formulation for individuals; individuals are treated as being fundamentally the same and with simple dynamics. Individually-oriented models (IOM), instead, model individuals in a way not possible with the continuous equations in IBM models, by adding discrete events, rules and more elaborate model structures. The IOM, therefore, involves more complex dynamics for the individual. Moreover, each individual in an IOM model can conceivably be modeled with different dynamics. In cases where the individuals are simple, as is generally the case towards the lower trophic end of the food web, an IBM, or even a general population model, may more reasonable, but for higher trophic level animals, such as birds, panthers and deer, an IOM may be more appropriate.

A key aspect of the OOPM methodology is that we can integrate each of these three modeling paradigms within the same logical structure. Lower trophic levels are generally modeled with a population model since there are too many individuals to make an individually-based model practical at this level. Moreover, the individuals have a sufficiently coarse type of aggregate behavior for which a population model is sufficient. Ulanowicz (1994) discusses this modeling level in some depth.

6.1 General Population Models

The lower trophic level within the Everglades has five key components:

1. Periphyton: micro-algae attached to a plant and bottom surfaces.
2. Macrophytes: submerged and emergent aquatic plants.
3. Detritus: dead organic matter.
4. Mesoinvertebrates: examples include water fleas and nematodes.
5. Macroinvertebrates: insect larvae.

Since *Periphyton* exists as a class in our conceptual model, we create a population object *periphyton* from class *Periphyton*. If we let the state variable (attribute) of *periphyton* be *pop*, and define two parameters for periphyton growth α and death β and neglecting predation, then we arrive at the object *periphyton* with the following attributes and methods:

- Attributes:
  - Variables: *pop*, α, β
  - Static Models: not defined.
- Methods:
  - Code: not defined.

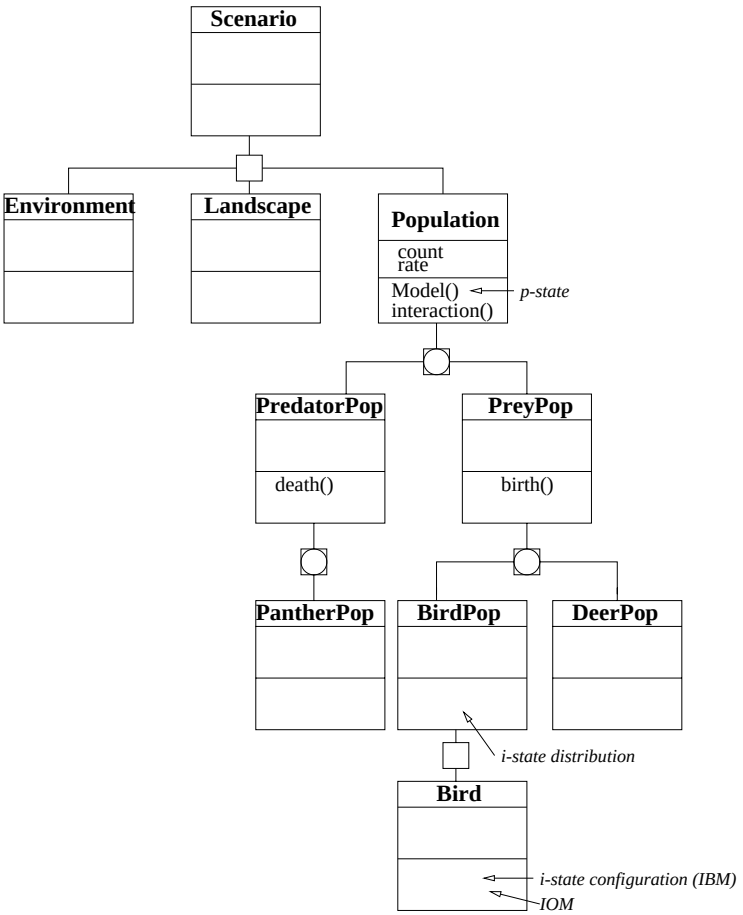

Figure 10: Lotka-Volterra population dynamics.

– Dynamic Models: (constraint-type) $\frac{d}{dt}pop = \alpha * pop - \beta * pop^2$

The dynamic model of *periphyton* specifies a simple population model, in the special case above, a logistic growth model. For many population models with competition for resources, the equations are only slightly more involved. For example, a Lotka-Volterra model would be constructed for two population objects *preypop* and *predatorpop*, each composed of particular sub-populations by constructing the necessary classes and objects. Such a model suggests that we have a physical scenario composed of an environment (weather), landscape and populations of organisms. There are two types of populations: predator and prey. For the sake of the biological metaphor, we choose *Panther* as the class of predator and *Bird* and *Deer* as sample prey classes. Fig. 10 illustrates the locations of other models that we will now discuss (i-state and IOM model types).

Let's note the rules for generalization and aggregation:

- Aggregation:

  1. A specific aggregation rule for attribute *count*:

$$Population.count \quad = \quad PredatorPop.count + PreyPop.count$$

A more general aggregation rule for *count*, keeping in mind updates and additions to this conceptual model, is:

$$C.count \;=\; \sum_i C_i.count$$

where C matches any class containing *count* and C_i matches the sub-classes of C.

2. An aggregation rule for dynamic model method *Model()* in *Population*:

$$\begin{aligned} PredatorPop.rate &= PreyPop.birth() - PredatorPop.death() \\ PreyPop.rate &= PreyPop.birth() - interaction() \end{aligned}$$

3. An aggregation rule for code method *interaction()* in *Population*:

$$interaction() \;=\; PreyPop.count \times PredatorPop.count$$

- Generalization: We let *count* be inherited (passed down) but not any of the methods defined in *Population*, *PredatorPop* or *PreyPop*.

In reviewing our model, we realize several important benefits from the use of OOPM in creating this model. The main benefit is one of knowledge representation that focuses on class creation and the lexical naming of equational terms such as *birth()* and *interaction()*. By making names explicit, we make the model more comprehensible. The benefits of structure passing are inheritance and aggregation. The definition of *Population.Model()* is invariant to additional *PredatorPop* or *PreyPop* sub-classes we may choose to add in the future. For example, we may later add *AlligatorPop* under *PredatorPop*. Since *AlligatorPop* would pass *count* upward via the first aggregation rule, the population model need not be redefined.

6.2 Structured Population Models

An important food source for wading birds, fish populations in the estuaries and freshwater marshes of the Everglades are modeled (Davis and Ogden 1994). Abiotic factors such as water level and salinity affect total fish biomass, as does the availability of resources. Mortality, such as caused by periodic or sudden drydowns, and predation reduce fish populations.

Various fish species or groups of species are divided into age- and size-class classes. Fish survival, growth, aging, and reproduction are modeled using continuous functions of the age- and size-class of species. Movement is also modeled as a function of age- and size-class and is species specific. For instance, movement of fish between mangrove swamps and interior marshes during periods of inundation is age- and size-specific for each species of fish. The hydrologic cycle is a principal ecological forcing variable and the total biomass for each class is updated.

Hallam et al. (1992) define a generic continuous structured model as being of the McKendrick-von Foerster type:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial a} + \frac{\partial}{\partial m}(\rho g) = -\mu(t, a, m, \rho)\rho \qquad (2)$$

DeAngelis and Rose (1992) use this type of structure for fish cohorts (groups). This model is slighly more complex than the general population model since time is not the only independent variable; a and m specify the age and biomass of the population. To "structure" a population by a physiological variable such as age or mass, it is necessary to partition that dimension. For example, if we are modeling the age a of fish, then we can structure along this dimension by discretizing age into integral ages (0,1,2, ...). A fish, for example, is in a larval stage or a adult stage. This provides a structural division of the age dimension into nominal values "larval" and "adult."

The method of incorporating structured models into the OOPM methodology is similar to the general population approach. Mass m, age a and density ρ are attributes of the population object. As in our Lotka-Volterra example, we can add new sub-classes of fish to a conceptual model without touching the dynamics. This sort of "plug and play" operation provides a significant level of flexibility in modeling and coding.

6.3 Individual-Based Models

Upper tropic level organisms (i.e., deer, panther, wading birds, and alligators) are modeled as individuals. Because deer are a primary food resource for panthers, these two species are combined and modeled in one sub-module. Their behavioral activities, such as movement across the landscape, are modeled in daily time-steps. The movement of deer is closely related to the spatial vegetation pattern and, apart from hydrological conditions, mainly determined by the occurrence and quality of forage the vegetation can provide. On the other hand wading birds movements, i.e. flights from their colony to their feeding sites and back, are almost completely determined by the hydrological patterns as they primarily forage in shallow water areas.

In contrast to deer and panther for which daily time steps are sufficient, wading birds operate on much smaller time scales. These time scales are determined by the various activities of the birds and can take several hours, e.g. for flying to a particular feeding site, or as little as several minutes for a bird that arrives at a feeding site with little or no prey and leaves almost immediately if feeding is poor. Wading birds are therefore simulated using an event queue.

The "events" in the wading bird model are specified by the various activities of the individual birds and are determined by behavioral rules. These rules can be divided into two sets: rules that determine what the bird is going to do after its current activity has terminated and rules that determine how a specific activity is performed. For example, after an adult bird has returned to its nest, a behavioral rule determines whether it will incubate eggs, or, if the eggs have hatched, whether the bird will feed its nestlings. Similarly, a rule in the latter set may be used to determine whether a bird must use flapping flight, thus using more energy, or may soar and glide which is more energy efficient. The behavioral rules are therefore either of the type:

IF (PREDICATE) THEN (NEXT_ACTIVITY_OF_BIRD)
IF (PREDICATE) THEN (HOW_TO_DO_SOMETHING)

In general, the predicate depends on the current—sometimes even some past—activity as well as the state or phase of the particular bird, e.g. its location. Some rules also take into

account activities of other birds of the same or of different species. For example, wading birds often forage in mixed-species flocks that then attract other wading birds to forage at the same location (Kushlan 1978). Rules that determine where a bird will forage must therefore take into account the foraging activities of other wading birds, i.e. their location.

The wading birds models execute using an event queue where the events are determined by the duration of the activities of the birds. A simple example is a flight from the colony to a feeding site where the duration of the flight is determined by the distance the bird between the colony and the feeding site and the bird's flight speed which in turn depends on the kind of flight the bird uses. Using a flight from a feeding site back to the colony as an example we can assign attributes and methods to an object *wading_bird* in much the same way as in Sec. 6.1. If we let one of the state variables (attributes) be the amount of energy the bird has stored in its body (e.g. as body fat) *fat* and define two parameters, the flight speed of the bird *speed* and the specific energy expenditure (i.e. energy expenditure per distance flown) *spec_en_ex*, then we arrive at the following attributes and methods for bird flight covering a distance d:

- Attributes:
  - Variables: *fat*, *speed*,*spec_en_ex*
  - Static Models: not defined.
- Methods:
  - Code: $time = d * speed$
  - Dynamic Model: Decrease *fat* by $d * spec\_en\_ex$

The dynamic model for a flight changes the state variable *fat* of the bird after the flight has been completed. The code method specifies the flight time *time* and therefore determines when the next event, its arrival at colony, is going to occur at which the bird makes a decision about its next activity such as relieving its mate from incubating the eggs or feeding their nestlings.

Various wading bird species can be incorporated into the OOPM methodology in the same as way as for general population and structured population models. Wood storks and white ibis, for example, have different attributes such as weight and size, exploit different food resources, and some of their behavioral rules are different. The "plug and play" operation is, however, slightly more involved than for the previous model types as some behavioral rules depend on the presence or absence of other wading species. Nevertheless, a new sub-class of wading birds, i.e. a new species, can be added to or deleted from the model without changing the conceptual framework.

7 Model Abstraction: Problems and Issues

The OOPM methodology and its implementation represent a beginning in tackling the model abstraction problem. Object Orientation, in general, even without the extension of *model*, provides for a convenient vehicle for talking about abstraction and dealing with its manifest issues. However, many problems remain to be solved if true model abstraction is to

be practical. Let's first consider what is possible in the implementation of OOPM. The crux of multimodeling is that it supports the hierarchical refinement of heterogeneous models through functional coupling. Each dynamic model is represented by the method (i.e., function) of a class. Within each model, there are the multimodel entry points defined in Sec. 5.3. These entry points differ for each model type. The entry points permit coupling in the traditional sense, thereby achieving a multi-level representation. Currently, MOOSE does not support type checking to ensure that a function's output type exactly matches the input type for another function's parameter; however, this problem is not an acute one.

Consider two dynamic models: A and B. Our methodology supports the basic case of B be defined as part of A, or vice versa. It also supports the idea of identifying a behavioral (but not necessarily structural) equivalent of B by using a neural network (Lee and Fishwick 1996) as the *black box* mechanism that captures the input-output relation to some arbitrary degree of accuracy. However, MOOSE does not support deriving A from B if A and B are independent. The derivability of models, computationally, is a much harder problem, and is discussed in more depth in (Zeigler 1984)(ch. 13) and (Fishwick 1995)(ch. 8). Ultimately, we would like to build a family of models, but it is not clear whether these models can be practically related since each model may have associated with it different underlying assumptions, state spaces and parameters. State and event spaces, for instance, should not be arbitrarily defined since they represent concepts in the minds of the modelers, and in the cognitive representations of the physical system. Our approach is not necessarily to "sweep the hard problems under the rug" but to provide a path to the greater goal of true model simplification. Heterogeneity in model creation and behavioral function simplication are stepping stones toward this goal. For a comprehensive survey of the state of the art in the model abstraction area, the reader is referred to a recent edited journal on the subject (Fishwick 1996c).

8 Conclusions

We have discussed a proposed method of multimodeling that has the potential to affect future ATLSS research. The attempt is to create a framework that helps with model design and model integration. The OOPM methodology has lead to a multimodeling implementation program called MOOSE (Multimodeling Object-Oriented Simulation Environment). MOOSE-created models (Cubert, Goktekin and Fishwick 1997) can be used by ecologists to visually design models which are automatically translated into computer code. This code can then be integrated into the existing ATLSS legacy code framework. However, we see OOPM and MOOSE as having the far more important effect of promoting model design and integration of models. A key problem, which we are beginning to address, is how to gradually phase-in visually structured models and phase out legacy code. This is a difficult problem. It is not enough to assume that legacy code will be abandoned, since that is impractical. Instead, we are looking into new approaches that foster a more gradual transition from legacy code to visual, object-oriented model.

The current state of MOOSE is that we have a working graphical user interface (GUI) for the conceptual model and a subset of the dynamic model types (FSA,functional block model). However, a considerable amount of time is being spent "cleaning up" the code to make it more robust and bug-free. While it is premature to speculate whether OOPM and

MOOSE will solve all problems with legacy code integration, we feel that it is a step in the right direction. Concerning new code development, the use of OOPM will be strongly encouraged from the beginning so that the code develops from a strong model specification. Code integration is always a difficult and time consuming task. Creating models within an object-oriented framework provides the modeler with much greater control and representation. OOPM goes beyond the software engineering visual designs by explicitly organizing knowledge about physical models. Even in the case where legacy code must be used without any code rewriting, we have determined that MOOSE can still be used to organize the legacy code even though no visual dynamic models are constructed initially.

Acknowledgments

We are grateful to two major groups: the MOOSE team and our ATLSS team partners. We would like to acknowledge the graduate students of the MOOSE team for their individual efforts in making MOOSE a reality: Robert Cubert, Tolga Goktekin, Gyooseok Kim, Jin Joo Lee, Kangsun Lee, and Brian Thorndyke. We also thank our colleagues on the ATLSS code development team Jane Comiskey, Don DeAngelis, D. Martin Fleming, Louis Gross, Michael Huston, Wolff Mooij, Phil Nott and Scott Sylvester. It is through this collaboration, that we have learned of the importance of integrating multilevel models and made the proposal that OOPM serve as a potential integration vehicle for the Everglades.

We would like to thank the following funding sources that have contributed towards our study of modeling and implementation of a multimodeling simulation environment for analysis and planning: (1) Rome Laboratory, Griffiss Air Force Base, New York under contract F30602-95-C-0267 and grant F30602-95-1-0031; (2) Department of the Interior under grant 14-45-0009-1544-154 and the (3) National Science Foundation Engineering Research Center (ERC) in Particle Science and Technology at the University of Florida (with Industrial Partners of the ERC) under grant EEC-94-02989.

REFERENCES

ATLSS 1997. Across-Trophic System Simulation (ATLSS): An Approach to Analysis of South Florida Ecosystems. Technical report, Biological Resources Division, US Geological Survey, South Florida/Caribbean Ecosystem Research Group, Miami, Florida.

Booch, G. 1990. On the Concepts of Object-Oriented Design. In Ng, P. A. and Yeh, R. T., editors, *Modern Software Engineering*, chapter 6, pages 165 – 204. Van Nostrand Reinhold.

Booch, G. 1991. *Object Oriented Design*. Benjamin Cummings.

Caswell, H. and John, A. 1992. From the individual to the population in demographic models. In DeAngelis, D. J.and Gross, L. J., editor, *Individual-based models and approaches in ecology*, pages 36–61. Chapman and Hall, New York, N.Y.

Cellier, F. E. 1991. *Continuous System Modeling*. Springer Verlag.

Cubert, R. M., Goktekin, T., and Fishwick, P. A. 1997. Moose: architecture of an object-oriented multimodeling simulation system. In *Enabling Technology for Simulation Science*, Orlando, Florida. Part of SPIE AeroSense 1997 Conference.

Dahm, C. N., Cumming, K. W., Valett, H. M., and Coleman, R. L. 1995. An Ecosystem View of the Restoration of the Kissimmee River. *Restoration Ecology*, 3(3):229–238.

Dale, V. and Rauscher, H. 1994. Assessing impacts of climate change on forests: the state of biological modeling. *Climate Change*, 28:65–90.

Davis, S. M. and Ogden, J. C. 1994. *Everglades: The Ecosystem and its Restoration.* St. Lucie Press, Delray Beach, FL.

DeAngelis, D. and Gross, L. J., editors 1992. *Individual-based models and approaches in ecology: populations, communities, and ecosystems.* Chapman and Hall, New York.

DeAngelis, D. L. and Rose, K. A. 1992. Which Individual-Based Approach is Most Appropriate For a Given Problem? In DeAngelis, D. L. and Gross, L. J., editors, *Individual-Based Models and Approaches in Ecology*, pages 67–87. Chapman and Hall, New York.

DeAngelis, D. L., Rose, K. A., B., C. L., Marschall, E. A., and Lika, D. 1993. Fish Cohort Dynamics: Application of Complementary Modeling Approaches. *The American Naturalist*, 142(4):604–622.

Fishwick, P. A. 1991. Heterogeneous Decomposition and Coupling for Combined Modeling. In *1991 Winter Simulation Conference*, pages 1199 – 1208, Phoenix, AZ.

Fishwick, P. A. 1993. A Simulation Environment for Multimodeling. *Discrete Event Dynamic Systems: Theory and Applications*, 3:151–171.

Fishwick, P. A. 1995. *Simulation Model Design and Execution: Building Digital Worlds.* Prentice Hall.

Fishwick, P. A. 1996a. A Taxonomy for Simulation Modeling Based on Programming Language Principles. *IIE Transactions on IE Research.* Accepted for Publication.

Fishwick, P. A. 1996b. Extending object oriented design for physical modeling. *ACM Transactions on Modeling and Computer Simulation.* Submitted for review.

Fishwick, P. A. 1996c. Guest Editor for Special Issue on Model Abstraction. *Transactions of the Society for Computer Simulation International*, 13(4).

Fishwick, P. A. 1997. A Taxonomy for Simulation Modeling Based on Programming Language Principles. *IIE Transactions on IE Research.* To be published in the latter part of 1997.

Fishwick, P. A., Narayanan, H., Sticklen, J., and Bonarini, A. 1994. Multimodel approaches for system reasoning and simulation. *IEEE Transactions on Systems, Man and Cybernetics.* to be published.

Fishwick, P. A. and Zeigler, B. P. 1992. A Multimodel Methodology for Qualitative Model Engineering. *ACM Transactions on Modeling and Computer Simulation*, 2(1):52–81.

Fleming, D., DeAngelis, D., and Wolff, W. 1995. The importance of landscape in ecosystem integrity: the example of Everglades restoration efforts. In Westra, L. and Lemons, J.,

editors, *Perspectives on Ecological Integrity*, pages 202–217. Kluwer Academic Publishers, Netherlands.

Forman, R. T. T. 1995. *Land mosaics: the ecology of landscapes and regions*. Cambridge University Press, Cambridge, Massachusetts.

Forrester, J. W. 1969. *Urban Dynamics*. MIT Press, Cambridge, MA.

Forrester, J. W. 1971. *World Dynamics*. Wright-Allen Press.

Fowler, M. and Scott, K. 1997. *UML Distilled*. Addison-Wesley.

Frohring, P. C., Voorhees, S. A., and Kushlan, J. A. 1988. History of wading bird populations in the Florida Everglades: A lesson in the use of historical information. *Waterbirds*, 11(2):328–335.

Graham, I. 1991. *Object Oriented Methods*. Addison Wesley.

Gunderson, L, H., Light, S., and Holling, C. 1995. Lessons from the Everglades. *Bioscience*. Supplement: Science and Biodiversity Policy S66-S73.

Hallam, T. G., Lassiter, R. R., Li, J., and KcKinney, W. 1992. Modeling Populations with Continuous Structured Models. In DeAngelis, D. L. and Gross, L. J., editors, *Individual-Based Models and Approaches in Ecology*, pages 312–337. Chapman and Hall, New York.

Halloway, M. 1994. Nuturing nature. *Scientific American*, 270(4):98–108.

Hill, D. R. C. 1996. *Object-Oriented Analysis and Simulation*. Addison-Wesley.

Hunsaker, C., Nisbet, R., Lam, D., Browder, J., Baker, W., Turner, M., and Botkin, D. 1993. Spatial models of ecological systems and processes: the role of GIS. . In Goodchild, M., Parks, B., and Steyaert, L., editors, *Environmental modeling with GIS*. Oxford University Press, Oxford, UK.

Kushlan, J. A. 1978. Feeding ecology of wading birds. In Sprunt, A., Ogden, J. C., and Winckler, S., editors, *Wading Birds*, pages 249–298. National Audubon Society, New York, N.Y. Research Report No. 7.

Kushlan, J. A. 1979. Design and management of continental wildlife preserves: Lessons from the everglades. *Biological Conservation*, 15(4):281–290.

Lee, K. and Fishwick, P. A. 1996. A Methodology for Dynamic Model Abstraction. *Transactions of the Society for Computer Simulation International*, 13(4):217–229.

Lima, S. and Zollner, P. 1996. Towards a behavioral ecology of ecological landscapes. *Trends in Ecology and Evolution*, 11(3):131–135.

Maxwell, T. and Costanza, R. 1995. Distributed Modular Spatial Ecosystem Modeling. *International Journal in Computer Simulation*, 5:247–262.

Murray, J. D. 1990. *Mathematical Biology*. Springer Verlag.

Norman, D. A. 1988. *The Design of Everyday Things*. Currency Doubleday, New York.

Ogden, J. C. 1994. A comparison of wading bird nesting colony dynamics (1931-1946 and 1974-1989) as an indication of ecosystem conditions in the Southern Everglades landscape. In Davis, S. M. and Ogden, J. C., editors, *Everglades: the ecosystem and its restoration*, pages 533–570. St. Lucie Press, Delray Beach, FL.

Peterson, J. L. 1981. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

Richardson, G. P. and Pugh, A. L. 1981. *Introduction to System Dynamics Modeling with DYNAMO.* MIT Press, Cambridge, MA.

Rosenberg, R. C. and Karnopp, D. C. 1983. *Introduction to Physical System Dynamics.* McGraw-Hill.

Rothenberg, J. 1989. Object-Oriented Simulation: Where do we go from here? Technical report, RAND Corporation.

Rumbaugh, J., Blaha, M., Premerlani, W., Frederick, E., and Lorenson, W. 1991. *Object-Oriented Modeling and Design.* Prentice Hall.

Stroustrup, B. 1991. *The C++ Programming Language.* Addison Wesley, 2 edition.

Thoma, J. 1975. *Bond Graphs: Introduction and Application.* Pergamon Press.

Ulanowicz, R. E. 1994. Modeling Lower Trophic Components of the Everglades Ecosystem. Technical Report UMCEES CBL 94-129, University of Maryland System, Chesapeake Biological Laboratory, Solomons, MD 20688-0038.

Ulanowicz, R. E. and Platt, T. 1985. *Ecosystem Theory for Biological Oceanography.* Canadian Bulletin of Fisheries and Aquatic Sciences 213, Ottawa.

Vasconcelos, M. and Zeigler, B. P. 1993. Discrete Event Simulation of Florest Landscape Response to Fire Disturbances. *Ecological Modeling*, 65:177–198.

Vasconcelos, M., Zeigler, B. P., and Pereira, J. 1995. Simulation of Fire Growth in GIS Using Discrete Event Hierarchical Modular Models. *Advances in Remote Sensing*, 4(3).

Wolff, W. F. 1994. An Individual-Oriented Model of a Wading Bird Nesting Colony. *Ecological Modelling*, 72:75–114.

Zeigler, B. P. 1976. The Aggregation Problem. In Patten, B., editor, *Systems Analysis and Simulation in Ecology.* Academic Press.

Zeigler, B. P. 1979. Multi-Level Multiformalism Modelling: An Ecoosystem Example. In Halfon, E., editor, *Theoretical Systems Ecology.* Academic Press.

Zeigler, B. P. 1984. *Multifacated Modelling and Discrete Event Simulation.* Academic Press.

Zeigler, B. P. 1985. Multifaceted Systems Modeling: Structure and Behavior at a Multiplicity of Levels. In *Individual Development and Social Change: Explanatory Analysis*, pages 265 – 293. Academic Press.

Zeigler, B. P. 1990. *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press.

Zeigler, B. P. and Moon, Y. 1996. Distributed Systems: Speed versus Error Tradeoffs. *Transactions of the Society for Computer Simulation*, 13(5):179–190.

Zeigler, B. P., Vasconcelos, M., and Pereira, J. 1994. Simulation of Fire Growth in Mountain Environments. In Price, M. and Heywood, O., editors, *Geographic Information Systems in Mountain Environments.* Taylor and Francis Publishing Company.

Biographies

Paul A. Fishwick is an Associate Professor in the Department of Computer and Information Science and Engineering at the University of Florida. He received the PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a senior member of the IEEE and the Society for Computer Simulation. He is also a member of the IEEE Society for Systems, Man and Cybernetics, ACM and AAAI. Dr. Fishwick founded the `comp.simulation` Internet news group (Simulation Digest) in 1987. He has chaired workshops and conferences in the area of computer simulation, and will serve as General Chair of the 2000 Winter Simulation Conference. He was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the *ACM Transactions on Modeling and Computer Simulation*, *IEEE Transactions on Systems, Man and Cybernetics*, *The Transactions of the Society for Computer Simulation*, *International Journal of Computer Simulation*, and the *Journal of Systems Engineering*. Dr. Fishwick's WWW home page is `http://www.cise.ufl.edu/~fishwick` and his E-mail address is `fishwick@cise.ufl.edu`.

James G. Sanderson is a Visiting Scientist in the Department of Wildlife Ecology and Conservation at the University of Florida. He received the PhD in Mathematics from the University of New Mexico in 1976. He also has 19 years of industrial/government research experience working on modeling and simulation at Los Alamos National Laboratory. His research interests are in ecological modeling and simulation, population modeling, and avian distributions. He is a member of the Ecological Society of America, and the Society for Conservation Biology. Dr. Sanderson's WWW home page is `http://www.cise.ufl.edu/~jgs` and his E-mail address is `jgs@cise.ufl.edu`.

Wilfried F. Wolff is a Visiting Scientist and Adjunct Professor in the Department of Biology at the University of Miami. He received the PhD in Physics from the University of Cologne, Germany, in 1981. He has held various visiting appointments at Stanford University, Xerox Palo Alto Research Center, University of Tennessee at Knoxville, and Oak Ridge National Laboratory. He is currently on leave of absence from the Juelich Research Center (KFA), Germany, where he is senior scientist in the Department of Biotechnology 3. His current research interests are in ecological modeling, in particular individual-based modeling, and simulation. He is a member of the Ecological Society of America and his E-mail address is `wilfried@fig.cox.miami.edu`.