International Journal of Geographical Information Science
Vol. 20, No. 9, October 2006, 965–990

Taylor & Francis
Taylor & Francis Group

Research Article

A probe mechanism to couple spatially explicit agents and landscape models in an integrated modelling framework

PHIL A. GRANIERO*† and VINCENT B. ROBINSON‡

†Department of Earth Sciences, University of Windsor, 401 Sunset Avenue, Windsor, Ontario N9B 3P4, Canada

‡Department of Geography, University of Toronto at Mississauga, 3359 Mississauga Road North, Mississauga, Ontario L5L 1C6, Canada

(*Received 3 October 2005; in final form 1 May 2006*)

Many environmental, ecological, and social problems require investigation using a mixture of landscape models, individual-based models, and some level of interaction between them. Few simulation-modelling frameworks are structured to handle both styles of model in an integrated fashion. ECO-COSM is a framework that is capable of handling complex models with both landscape and agent components. Its Probe-based architecture allows model components to have controlled access to the state of other components. The ProbeWrapper is a modification of this common design approach which allows alterations to the state retrieved from the model and is a critical component of ECO-COSM's broad modelling capability. It allows agents to apply perceptual filters or measurement errors to their observations of the landscape, or apply decision-making strategies in the face of incomplete or uncertain observations. ECO-COSM is demonstrated with a landscape model of metapopulation dynamics, an agent model of squirrel dispersal, and a coupled landscape-agent model to evaluate field-data-acquisition strategies for identifying nutrient or contaminant hotspots.

Keywords: Object oriented; Agent; Landscape model; Individual-based model; Simulation framework

1. Introduction

Parker *et al.* (2002) review a large body of research that uses models of multiple autonomous agents making land-use decisions by using information about the landscape, its relation to the agents, and interaction among the agents themselves. Their decisions in turn influence the evolution of the underlying landscape. The agents themselves do not necessarily have a spatially explicit representation. As land-use managers, the agents' decisions regarding land-cover change are not usually dependent upon their respective locations on the landscape when they make their decisions.

Knowledge of the agents' locations upon the landscape is essential in other modelling domains, such as in studies of animal or human movement (e.g. Batty *et al.* 1998, South 1999, Westervelt and Hopkins 1999, Kukla *et al.* 2001). Typically, these agents are not primary drivers of landscape evolution; rather, the landscape

*Corresponding author. Email: graniero@uwindsor.ca

International Journal of Geographical Information Science
ISSN 1365-8816 print/ISSN 1362-3087 online
http://www.tandf.co.uk/journals
DOI: 10.1080/13658810600830541

drives the agents' future states and spatial arrangement. Although the landscape is slowly evolving under the influence of larger scale forces, the landscape is treated as a static pattern during the short modelled time frame. These models are generally constructed using an agent-based modelling framework that can access GIS data. Brown *et al.* (2005) outline methods and alternative architectures for coupling agent model development platforms and GIS along with several examples. Using more illustrative examples, Westervelt (2002) discusses several theoretical and technical issues involved in coupling agent models with GIS.

There are problem domains where the temporal scale of landscape change is similar to the scale of agent movement and decision-making, for example reaction and emergency response during a natural hazard such as a fire or flood. In these cases independent, complex agent models and landscape models may operate simultaneously, with mutual influence on the other's future state. Holt *et al.* (1995) and Westervelt and Hopkins (1995) are early, problem-specific examples of linked agent-landscape models. The ATLSS system (DeAngelis *et al.* 2000, Gross and DeAngelis 2001) is one of the largest projects coupling individual-based models, landscape and process models, and GIS data.

A modelling framework is a reusable, 'semi-complete' application that can be extended or specialized to produce custom applications (Johnson and Foote 1988), speeding the development process but not necessarily removing programming or the need to understand programming (Inchiosa and Parker 2002). Swarm (Minar *et al.* 1996) is one of the earliest and best-known general-purpose agent-based modelling frameworks where, in the original version, the landscape was a sub-swarm of 'landscape cell agents'. More recent versions include grid components, and GIS connections are being considered. Echo (Holland 1992, 1994) uses a schematic representation of grid 'sites' to model species abundance (Forrest and Jones 1994) and community structures (Hraber and Milne 1997) under different interaction behaviours. Gecko (Booth 1997), which is now incorporated into Swarm, extended Echo's spatial structure in a more geometric sense and incorporated more explicit spatial constraints into agents' movement to model species at multiple trophic levels within the landscape (Schmitz and Booth 1996).

ASCAPE (Parker 2001) is one of the first in a new generation of powerful agent-based modelling frameworks. It has a highly interactive environment that allows a user to control the model dynamics while the model executes. It requires some programming skill, and its more abstract range of world topologies tend to be less suited to spatially explicit landscapes. MASON (Luke *et al.* 2005) is a framework that can support millions of agents with a rich range of visualization components for a variety of topological spaces. It supports a range of 2D and 3D spaces, but none are explicitly matched to GIS data models. Repast (Collier *et al.* 2003) has emerged as one of the more powerful and popular frameworks for constructing agent-based models to explore highly complex social interactions. It directly imports GIS data, including ESRI shapefiles and ASCII raster files. However, its agent feature set can generate a steeper learning curve for developers who only require simple agent structures in their modelled environment.

Object-oriented design (OOD) forms the basis of most modelling frameworks that are noted above. This process of decomposing a system of interest by hierarchical abstraction (Booch, 1994) has become quite familiar and is a mainstay of the agent-based ecological modelling community. The OOD community's collective experience has identified several *design principles* that help a designer choose between

alternative decompositions of a problem (Martin 2003). Similarly, certain common abstraction structures emerge across problem domains and systems. *Design patterns* are meta-abstractions that formally document problem scope, general solution structure, alternative forms, and common implementation issues (Gamma *et al.* 1995). They form both a toolbox of solutions for common design problems and a vocabulary that helps object-oriented designers to discuss design alternatives. The common design pattern vocabulary is used wherever possible in this paper to describe general design principles or software structure.

This paper illustrates how the range and flexibility of a simulation modelling framework may be augmented by specifically exploiting the Principle of Indirection and the associated *Delegation* and *Decorator* design patterns (Gamma *et al.* 1995). This is illustrated using ECO-COSM as an example of a framework which can accommodate general landscape models, spatially explicit agent-based models, and coupled agent-landscape models within a GIS framework using a small class library. In section 2, ECO-COSM's motivation and goals are outlined and compared with other common frameworks. Its architecture and subsystems are described with a focus on the Probe structure. An example model illustrates a dynamic landscape with static observers. Section 3 describes how the Probe structure was exploited to create support for mobile, spatially explicit agents on a static landscape. Section 4 demonstrates the early steps of developing a simple dynamic landscape coupled with dynamic observers. Section 5 concludes with some summary discussion.

2. ECO-COSM framework

ECO-COSM (Extensible Component Objects for Constructing Observable Simulation Models) provides a library of modular software objects that manage the structure of space and time within a simulation, including mechanisms to handle concurrent activity among objects within the simulation. The framework is freely available for download at http://matrix.memf.uwindsor.ca/projects/eco-cosm. This section provides a conceptual overview of the basic components in the modelling subsystem. Full details of the ECO-COSM structure are described in Graniero (2001) and in the reference documentation available on the website.

ECO-COSM was originally conceived to explore the interaction between field sampling strategies (such as spatial pattern, density, and measurement accuracy) and our ability to characterize complex ecological systems (Graniero 2001). It focuses on simulation experiments where the simulation trials are executed as independent programs and parameterized with command-line arguments, or as parameterized system objects in a larger simulation experiment program. A simulation may take only seconds or minutes to execute, but in an experiment several thousand trials may be executed across varying landscapes, with hundreds of alternative data sets drawn from each trial. The design therefore was motivated by data extraction and analysis rather than real-time visualization and interaction. Several design goals guided ECO-COSM development:

- highly flexible in extracting and manipulating localized state;
- easily interchangeable spatial structures and spatial interactions;
- usable by students and scientists with minimal formal training in programming;
- a small, simple core which can be extended for specific applications;
- access to spatial data stored in various GIS formats;

- easily embedded into simulation control programs or scripts; and
- visualization tools separated from simulation tools.

It was originally designed to support coupled lattice landscape models, but as we will demonstrate in section 3, its architecture allowed the addition of a basic agent subsystem with about 2 weeks' worth of development effort.

We first examined the possibility of extending Swarm, one of the better-known simulation frameworks at the time. However, the use of an uncommon language (Objective-C) available only on limited platforms was felt to be too large a barrier to general portability, and an impediment to effective use by less experienced programmers. ECO-COSM was written in Java using its most general language features and to date has been robust on a range of computer platforms and virtual machines. Java was selected as the implementation language for the same reasons commonly described by others (e.g. Wood 2002, Tobias and Hofmann 2004, Weidmann and Girardin 2005). It is commonly understood that Java is slower than fully compiled languages, but we contend this is less of an issue than in the past. Newer Java Development Kits (JDKs) are faster than their recent predecessors by a factor of two (Weidmann and Girardin 2005), and processing speeds have increased dramatically even in inexpensive computers. Execution time is traded off against the ease of developing and testing the model, and human time is generally more valuable than computing time.

2.1 *System structure*

At the highest level, ECO-COSM has five main subsystems: Scheduling, Modelling, Instrumentation, I/O, and Utility. The simulation model proper is constructed using components from the Modelling subsystem, which control the spatial and temporal structure of the model. During execution, components from the Scheduling subsystem trigger events within the model. Optionally, components from the Instrumentation subsystem may be used to extract data from the operating model. The instrumentation is also triggered by the Scheduling subsystem. Components in the Utility subsystem include parameter sets, random distribution generators, and statistics collectors which are used throughout the framework to assist with many tasks. The I/O subsystem includes components for model management, data file manipulation, and formatted data streams including GIS sources.

Each simulation program has one instance of the Clock and the Schedule from the Scheduling system. Any object may access the Clock's time. The Schedule keeps track of all pending actions (ScheduleItems) and triggers items based on scheduled time and a Precedence code, which may be used to enforce order on types of actions scheduled for the same time. The Scheduling subsystem is an event-driven structure (Banks and Carson II 1984). The Schedule advances the Clock time to the next pending ScheduleItem and executes that action by invoking the *execute( )* method of the embedded Command object. The pattern is repeated until a termination action is encountered or nothing remains to be done on the Schedule. A ScheduleItem may be configured so that once it is executed, it reschedules itself a set time interval into the future. Iterative time-step models like cellular automata or coupled lattices use this repetition feature to continually reschedule their calculation actions for the next time step.

The abstract Simulation class is the central component of any modelling exercise (figure 1). The programmer extends the class and writes a *setup( )* method that

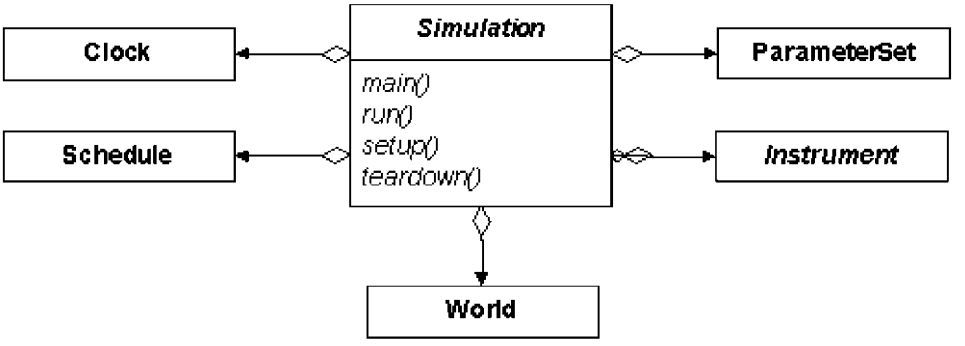

Figure 1. Simplified UML diagram showing the main subsystems and objects of the ECO-COSM simulation framework. Complete, detailed UML diagrams are available at http://matrix.memf.uwindsor.ca/projects/eco-cosm.

structures the simulation for the desired model, attaches instrumentation, and acquires memory or file resources required for the model. The *run( )* method is very simple and is the same for all concrete Simulations: until the Schedule is finished (no more pending items or a terminate command was triggered), trigger the next pending item on the Schedule. The programmer extends the *teardown( )* method to finish any data collection, release resources, and get ready for program termination.

A Simulation object is composed of many other objects: references to the Clock and Schedule to control the model execution; a ParameterSet (described below) to set up the particular run-time conditions for the model; a World (section 2.2) which is the simulation model proper; and possibly one or more Instruments (section 2.3) to collect data from the operating simulation.

A Simulation class can invoke a model in two ways: as a stand-alone program invoked by the user from the command line via the *main( )* method; or as an object within a larger model management application. The *main( )* method generally executes the following steps: (1) create a ParameterSet from the command line; (2) create an instance of itself using the ParameterSet for argument values; (3) call *setup( )*, *run( )*, and *teardown( )* in order; and (4) terminate the program. A model management application would generally: (1) create a ParameterSet using its own internal functions; (2) create an instance of the Simulation object using the ParameterSet for argument values; (3) call *setup( )*, *run( )*, and *teardown( )* in order; and (4) carry on with its own functions. A ParameterSet is a collection of key/value pairs used to configure the model. Some convenience methods determine if values are present for a specified list of keys, and construct key/value pairs from a collection of '*name=value*' strings from the command line, a configuration file, or other source.

2.2 *Modelling subsystem*

An ECO-COSM simulation contains a single World object, which manages the overall spatial structure and mediates communication between components of the simulated world (figure 2). The World is composed of one or more Layers, each identified by a keyword and representing some attribute or collection of features that are of interest in the simulation. This structure matches the conceptual structure of most GIS packages. Specification of processes and behaviour in the modelled

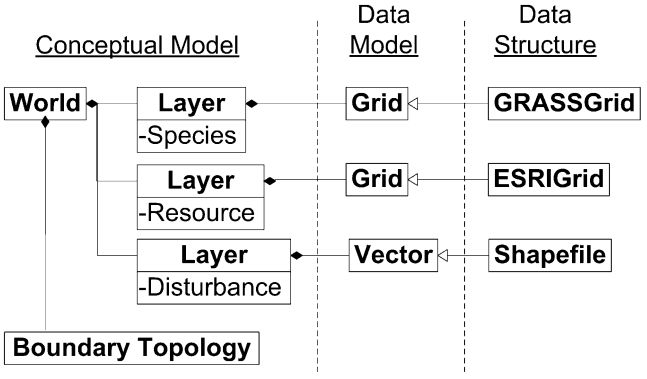

Figure 2. Simplified UML diagram of the main objects comprising a simulated world within the modeling subsystem. Note that the levels of abstraction in the design match with the conceptual model, spatial data model, and spatial data structure in GIS design.

world is done at the high level of 'resource layer', 'land cover layer', or 'disturbance zone' without the distraction of how the state is extracted from the GIS.

When a Layer is instantiated, it is associated with a StepRule object. The StepRule structure follows Martin's (1996) Dependency Inversion Principle (i.e. do not depend on concrete classes; depend on abstractions), which is a fundamental design principle in most frameworks. The StepRule is a Java *interface*, a purely abstract declaration of method signatures, preferably with documentation describing the intended purpose of each method. StepRule declares a single method called *step( )* which calculates the next state of an object within the simulation model. A class can declare that it *implements* an interface if it has a concrete implementation of all methods declared in the interface; a *step( )* method in the case of a StepRule. Instances of the implementing class may be used anywhere a StepRule type is expected. When a StepRule object is instantiated, a target Layer is named as an argument. When a Layer is expected to calculate its next state, it calls its StepRule's *step( )* method, and the calculations are carried out. In the case of a grid representation, the StepRule will establish the next state of each grid cell. The binding gives the StepRule private access to the Layer's state through methods such as *getCurrentState( )* and *setPendingState( )*. If the Layer is static, a NullStepRule is used, which carries out no actions in its *step( )* method. In order to change the dynamics of a Layer, a different kind of StepRule is assigned to it. Additional complexity is added to a basic model by creating new, more sophisticated concrete StepRule implementations. This is where the bulk of ECO-COSM model design and code development takes place.

Referring to Layers by using keywords is an application of the Principle of Indirection (i.e. reference something using a name, reference, or container instead of the value itself). If the next state depends in part on another Layer, the StepRule may send a request to the World in the form *getState(layername, location)* where *layername* is a keyword associated with a Layer, and *location* is a Location object containing (x, y) coordinates. The StepRule is not concerned about the representation or structure of the Layer. As long as the conceptual model specifies that such a Layer will exist in the World, the StepRule may be designed to consider the state in that Layer when making its calculations.

The Layer is an abstract representation, aligning with the developer's conceptual model of the landscape. It must be associated with an implementation of the DataModel interface to manage the general mechanism for accessing the correct value at the specified location. For example, if an object queries the World for the 'resource' state at a given location, it ultimately is not concerned whether the value is stored in a grid cell at that location (i.e. a GridModel is used), or stored as the attribute value in a polygon containing that location (i.e. a VectorModel is used). Although the DataModel object manages the mechanism, it does not access the actual data. A DataModel must specify a DataStructure, which has the code to query a particular spatial database format and return a value. The DataStructure is also an interface; concrete implementations of the grid data model include GRASSGrid, ESRIGrid, and ECOCOSMGrid. The modeller may choose to change an input layer from a GRASS raster to an ESRI shapefile, and the StepRule representing the modelled system's dynamics will not have to change. Of course, the modeller must still consider the implications of changing the underlying spatial structure of a reference Layer.

The World must be associated with a BoundaryTopology implementation, which sets the policy that describes how edges will be handled within the simulation, and therefore how the (x, y) coordinates in a Location object map to the 'physical' extent of the World (with respect to the spatial database). This is done by invoking another aspect of the Principle of Indirection via the *Delegation* design pattern: that is, one class 'hands off' responsibility for a task to another class, possibly augmenting the request as it is forwarded. When a simulation object makes a request for the state of a Layer at a particular Location, it is done at the conceptual level of the model. The location is chosen with regard to the general logic that makes sense within the modelled world, and not with regard to the particular data set underlying the executing model. Before querying a target Layer, the World passes the Location to the BoundaryTopology which returns a new Location or indicates an error if the logical value violates the rules which govern the mapping to the World's 'physical' extent. The World actually queries the Layer with the transformed Location. Under normal conditions, the original and transformed Locations are identical. At the edge of the World's 'physical' extent, the transformed Location depends upon the specific BoundaryTopology implementation.

Consider an implementation of the well-known model Conway's Game of Life (Gardner, 1971). The World's edges are at (0,0) in the southwest and (99,99) in the northeast. The World is composed of one Species Layer using the Grid DataModel, 100 by 100 cells in size. The state of a cell is either 1 (alive) or 0 (dead). The next state of each cell in the Species Layer depends on the current state plus the state of all immediately neighbouring locations. The next state for location (50,99) must be evaluated: what is the value at (50,100)? A HardBoundary implementation would report an 'out of bounds' error, and the StepRule would react in the manner that the modeller specified for such cases (in this case, ignore the cell). A WrappingBoundary implementation would do a modular transformation of the location, and the World would actually return the state at (50,0), effectively creating a torus. By substituting different BoundaryTopology implementations and executing the otherwise unchanged model, the modeller can explicitly examine the effect of edges on model results (figure 3). The model's behavioural specification has not changed, so any difference in result is solely caused by the way edges are treated.

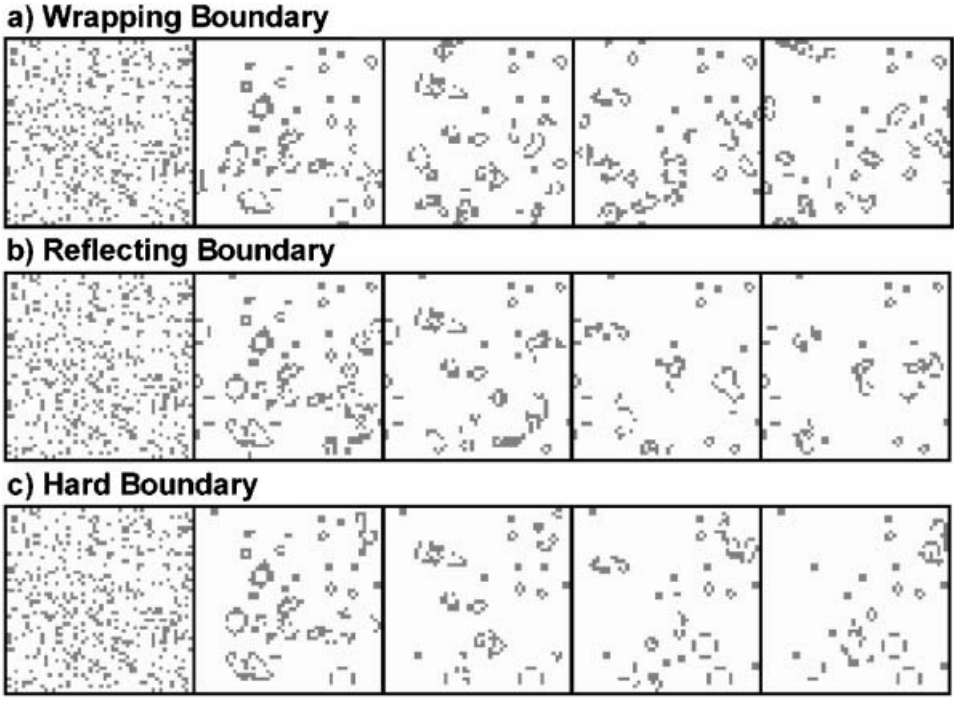

Figure 3. Edge effects on Conway's Game of Life (Gardner 1971). 50 by 50 cell grids at time 0, 10, 20, 30, and 40. The same ConwayLife StepRule and initial grid configuration was used in each example, only the BoundaryTopology differs. All changes in pattern can be attributed to the change in handling references to cell locations outside of the grid's physical extent.

Abstraction of the conceptual model—data model—data structure chain, abstraction of locations within the world, and abstraction of the boundary topology are very powerful from the standpoint of model development. However, the presence of multiple levels of abstraction, the potential for heterogeneous data sources, and a variable policy for coordinate transformation that is unknown a priori mean that the mechanism to access state within the operating model must also be abstract. ECO-COSM therefore has an instrumentation subsystem that provides abstracted data access.

2.3 *Instrumentation subsystem and Probes*

The instrumentation subsystem has few classes but is conceptually the most complex and most versatile in the entire framework (figure 4). It allows the modeller to create very sophisticated data-collection tools, connecting them to a working simulation model with no impacts or side effects on the model calculations. A basic model will execute successfully from start to finish and terminate with none of the results observed or captured. The model, representing the formalized behaviour of some ecological system, remains completely separate from any results data that are gathered for later analysis. Get/set methods are a common idiom used by several frameworks, including ASCAPE, to control access to state within model elements. This is good practice in general OO programming for automatic, extensible access via introspection which reduces the need for explicit coding. However, one can

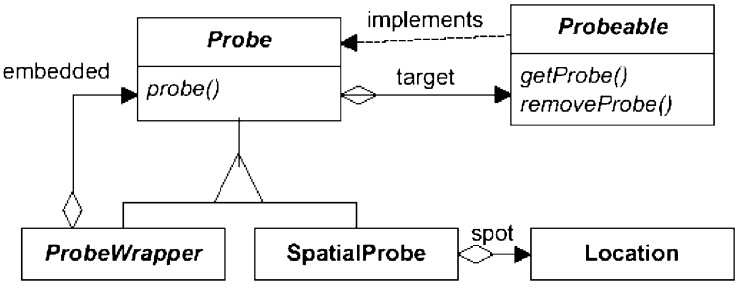

Figure 4. Simplified UML diagram of the Probe, Probeable, ProbeWrapper relationship (from Robinson and Graniero 2005a).

accidentally expose too much access (especially editing access) to elements of a simulation model, which increases the risk that instrumentation and user controls may incorrectly affect model state.

In ECO-COSM, access to any get/set methods is restricted to low-level objects belonging to the modelling subsystem. Instead, model objects such as StepRules must obtain Probes from Probeable model components, providing access to a component's state but keeping the mechanics opaque. A Probe must be explicitly written to give read-only access to particular state. Instruments must request a Probe, and the model results are insulated from potential programming side effects due to changes in monitoring or data-collection code. This common *Observer* design pattern (Gamma *et al.* 1995) forms the basis for the Model-View-Controller (MVC) approach to graphical user interface design (Krasner and Pope 1988), and is used by Swarm Probes, ASCAPE Views, and MASON Inspectors to access and graphically display model state during execution. ECO-COSM Probes may also be used in this way, but to date, our priority has focused on data extraction and analysis, and there are few Probes for visualization. New Probes for graphical views may be created when needed.

The primary differentiating feature of the instrumentation subsystem lies in the ProbeWrapper, a specialization of the Probe interface and an implementation of the *Decorator* design pattern (Gamma *et al.* 1995). Every ProbeWrapper instance has another Probe (possibly a ProbeWrapper itself) embedded within it. When the ProbeWrapper is queried, it in turn queries the embedded Probe. When it receives the embedded Probe's result, the ProbeWrapper may perform any kind of operation on it before passing it on as its own result. In this manner, the 'pure' state returned by a Probe may be modified: the modeller may work with data that are derived from the model's current state but do not necessarily match the exact state. The power of this approach is illustrated in the example below.

2.4 *Landscape model example*

ECO-COSM was used to construct a spatially explicit, patch-occupancy metapopulation model that extends the original model developed by Caswell and Cohen (1991), integrating resource heterogeneity into the system's colonization, extinction, and competitive pressure processes (figure 5). Local resource availability depends upon a static *substrate* Layer b and a global climate signal time series $M(t)$. The value of each cell n in the *resource* Layer r at a given time t is calculated over a range

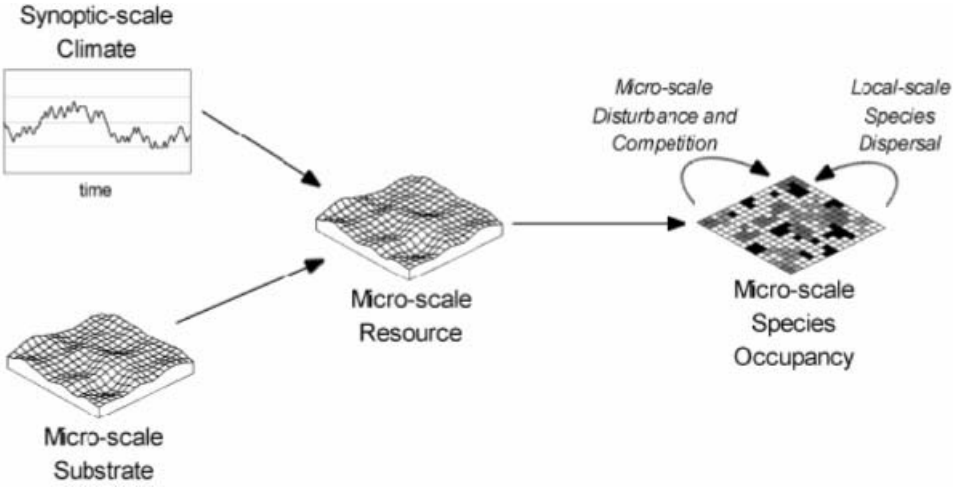

Figure 5. Spatially explicit patch occupancy model incorporating stochastic CA-style species interactions. Local colonization, extinction and competition parameters depend upon local resource levels, which in turn are dependent upon a static substrate, a global climate signal, and regional disturbances (from Graniero 2001).

[0–1] as:

$$r_{nt} = M(t) \times b_n \tag{1}$$

The model includes two interacting species: a 'fugitive' species which can easily colonize new areas, and a 'superior' species which is not as resilient but can eliminate the fugitive species via competitive exclusion. In *species* Layer *f*, each cell *n* indicates which of the species, s_1, s_2, or both, occupy the cell. At a given time *t*, the probability D_i that species s_i will be disturbed out of an occupied cell *n* is

$$D_i = p_{di}^{r_n} \tag{2}$$

where p_{di} is a resiliency coefficient for species *i*. The probability C_i that species s_i will colonize an unoccupied cell *n* is

$$C_i = 1 - e^{-f_i d_i r_n} \tag{3}$$

where d_i is the dispersal coefficient, and f_i is the frequency or proportion of local cells (examined over a specified kernel) occupied by species s_i. The probability of competitive exclusion of species 2 by species 1 is p_c. These probabilities were combined to form a probabilistic state transition matrix (table 1). The state

Table 1. Probabilistic state transition matrix for the patch occupancy model[a].

		From state			
		0 (−)	1 (s_1)	2 (s_2)	3 (s_1, s_2)
To state	0 (−)	$(1-C_1)(1-C_2)$	D_1	$D_2(1-C_1)$	D_1D_2
	1 (s_1)	$C_1(1-C_2)$	$(1-D_1)$	C_1D_2	$(1-D_1)(1-D_2)p_c+(1-D_1)D_2$
	2 (s_2)	$(1-C_1)C_2$	0	$(1-C_1)(1-D_2)$	$D_1(1-D_2)$
	3 (s_1, s_2)	C_1C_2	0	$C_1(1-D_2)$	$(1-D_1)(1-D_2)(1-p_c)$

[a] s_i within the parentheses associated with a state indicates that species *i* is present in that state.

transitions were embedded into StepRules which are associated with their respective Layers; at each time step, each Layer's StepRule is invoked to calculate the next state for each cell. As the model is being set up, a Command object that invokes a target Layer's StepRule is placed on the Schedule for each Layer and configured to automatically reschedule itself for the next time unit.

The patch occupancy model was used as a surrogate landscape to investigate the influence of spatial sampling strategies on the accuracy of characterizing an ecosystem (Graniero 2001). A subset of the grid cell 'patches' were sampled at set time intervals with respect to their resource level and the species occupying the patch, and the results were written to a data file. For each sample patch, a Probe targeted at the patch's Location was obtained from each Layer. This simulated a network of sensor stations measuring resource level, or a field worker revisiting sample sites to take a measurement and identify species sighted during the visit. The value recorded at each location was not necessarily identical to the model state: random Gaussian noise was added to the resource level to simulate instrument error, and the species state was randomly changed for some proportion of locations to simulate classification error. The ProbeWrapper mechanism to simulate this field-data collection is presented in figure 6. The sample data were then used to estimate macro-scale system characteristics at each sample time. A total resource inventory was estimated using a representative area-weighted sum of the measurement values across the entire study extent. A species census was estimated as the frequency of sampled patches occupied by each species.

While sample data sets were being gathered at run time, other Probes captured the state of the entire Layer without modification by ProbeWrappers. Utility classes added up the *resource* Layer cells to produce the 'true' resource inventory and calculated the 'true' frequencies of cells occupied by each species in the *species* Layer. By examining how closely the estimates described the actual state of the running model, it was possible to evaluate the performance effects of different sampling design choices. For example, does the choice of spatial pattern (e.g. random, uniform) matter? Given a choice between reducing measurement error or increasing sampling density, which one generally reduces the magnitude of error or degree of bias in the estimates?

In the experiment, the model was executed a large number of times using randomly generated landscapes governed by different substrate patterns: (1) uniform value, (2) gradients of varying degrees, and (3) patches with varying

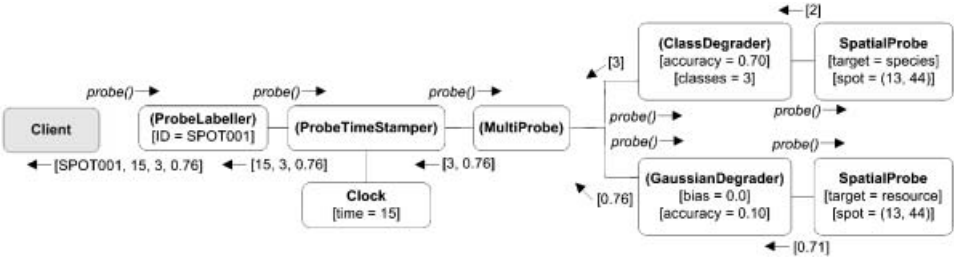

Figure 6. Example of a nested Probe composition used by an Instrument. Names in parentheses indicate ProbeWrappers. The state of two Layers, namely *species* and *resource*, are probed for their current state at location (13,44). The values are degraded to simulate instrument and classification error, and the results are combined together into a single time stamped record.

complexity and variability. For each model execution, 250 different field collection campaigns were simultaneously conducted, with sampling strategies governed by (1) Random or uniform (gridded) pattern; (2) spatial sampling density varying between 0.04% and 4% of grid cells; (3) instrument error varying between 0% and 30%; and (4) species identification success rate varying between 100% and 60%. Once the simulations were completed, the resource inventory and species census were estimated for each time in each data set, and the estimates were compared with the actual values calculated from the running model and logged to a file. The comparisons from all of the model trials were aggregated for each sampling strategy. The error and bias distributions were analysed for trends based on sampling characteristics.

In general, the results showed that the choice of spatial pattern affected the resource inventory but not species census, regardless of species rarity. For inventory, when trend identification (e.g. increase, steady, decrease) is important, one should use a gridded pattern to control magnitude of error, but when identifying health and safety thresholds is important, one should use a random pattern to minimize bias. With respect to density versus accuracy, the general results were the same for both resource inventory and species census: at densities we can realistically achieve in the field, increasing spatial density almost completely drives the improvement in macro-scale estimates, and the accuracy of individual measurements has almost no influence. From the standpoint of field planning, the results indicate that resources should be chosen for rapid, dense coverage rather than *n*th decimal accuracy for these macro-scale descriptions: mobile computers and GPS to speed up collection and recording; equipping more field workers with inexpensive (but reliable) sensors for resource inventory; and enlisting more field workers with basic training for species census (e.g. the Breeding Bird Survey). It is important to note that these results apply only to macro-scale characteristics like total inventory or total census with patch-level observations. More simulation studies must be conducted to evaluate the effect on estimating other ecosystem characteristics, or the effects in other types of ecosystems.

In a real-world study represented by this simulation experiment, the data values are typically gathered by a team of field workers moving through the environment. This is similar to Grimm and Railsback's (2005) notion of a 'virtual ecologist'; that is, the modeller programs an ecologist observer into an ABM framework that samples the underlying landscape model in the same way that the real ecologist would observe the natural system. However, in this study, the amount of change in the environment during a particular survey session was assumed to be minimal, and the order and timing of the individual measurements could be ignored. Therefore, the extra overhead of using mobile agents was not necessary for the study, and as such, static probes could simulate the 'virtual ecologists'.

3. Agent-based models and ECO-COSM Probes

Any computing environment designed to support development of spatially explicit individual-based models must allow the agent to evaluate and interact with other individuals as well as to acquire and maintain knowledge about the surrounding landscape (Westervelt 2002). From a GIS perspective, an animal agent must be able to query the state of relevant GIS layers within its local perceptual range and use that information to make decisions regarding its mobility, behaviour, and change of internal state (figure 7). Within the ECO-COSM framework, this requirement is

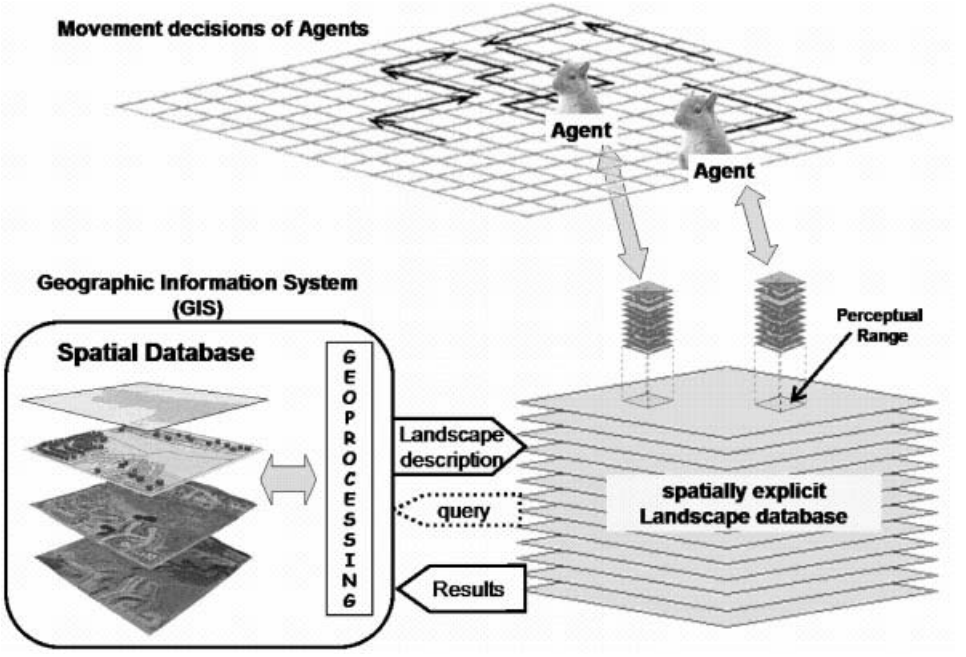

Figure 7. Conceptual illustration of the major components of a spatially explicit ecological model that focuses on movement behavior of individual animals, e.g. natal dispersal. Note the loosely coupled relationship with the geographic information system (from Robinson and Graniero 2005b).

directly handled by the Probe mechanism. By obtaining Probes from relevant Probeable landscape layers (and possibly from other agents as well), an agent can acquire a perceptual inventory of its world. This fits Bian's (2003) hybrid approach to representing the world in individual-based modelling, which incorporates a traditional grid model of the environment and an object-oriented model of individual organisms. More sophisticated field sampling simulation experiments need mobile agents, and the Probe mechanism already provided the necessary foundation for an agent to view its world and make movement decisions. Thus, the Modelling subsystem was extended to include basic structures for constructing agent-based models.

The Agent interface provides a purely abstract specification of the methods or messages to which an implementation must respond. A particular Agent instance's movement and decision-making behaviour are defined by a StepRule instance assigned upon initialization. The World can include zero, one, or more Population objects, which are collections of Agents. Populations do not have hierarchical agent behaviours of their own like in MASON, Swarm, or ASCAPE. The primary focus for ECO-COSM is modelling the interaction of a small number of independent mobile agents with their surrounding landscape. Some communication between agents may happen, but hierarchical social structures are not prevalent in the domain of interest. Populations are general collections with some convenience methods much like a ScapeVector in ASCAPE. When a Population is told to *step( )*, it iterates through its Agents, invoking their respective *step( )* methods. Agents can be moved from one Population to another, for example when their state changes from 'active' to 'dead'. Broad model behaviour can be triggered based on

Population conditions such as terminating the model when the 'active' Population's Agent count hits zero.

Agents perceive elements of their environment by obtaining Probes from the Layers or Agents of interest. An important principle in individual-based modelling is that each Agent may view and respond to the world in a different manner. This is easily achieved by using ProbeWrappers, which in essence are used to somehow modify the 'pure' result retrieved from a Probeable object. For example, the land cover type observed at a distance may be subject to random misclassification due to limits of perceptual range. Alternatively, the state's description scheme may be modified to suit the purpose of the observer: the grid cell may be described as 'mature oak' in the land cover Layer, but the observing Agent may simply perceive it as 'trees'. By wrapping Probes in slightly different ways for different individual Agents of a certain type, it is possible for the modeller to introduce variation in an individual's ability to perceive the world while using the same basic decision-making process.

3.1 *Agent model example*

ECO-COSM was used to explore the use of fuzzy logic in an individual-based model of natal dispersal behaviour of eastern grey squirrels (*Sciurus carolinensis*) in an area near the Land Between the Lakes National Recreation Area (Robinson and Graniero 2005a). The motivation for this exploration derives from the problems of uncertainty in both data and model parameters when using GIS-based spatially explicit population models. A key component in most spatially explicit population models is the process of dispersal, the mechanism by which species spread and establish new sub-populations. However, uncertainties in the spatial data as well as the model parameters can have an important effect on the performance of these models. Robinson (2002) sketched out, in theory, how fuzzy logic may be useful in addressing the persistent problem of uncertainty in such spatially explicit models. Using a well-known species with strong natal dispersal behaviour, this modelling exercise (1) explored issues related to the incorporation of fuzzy logic in the control of individual object behaviour, (2) developed a plausible, prototypical fuzzy-logic-based model of individual dispersal movement, and (3) compared the behaviour of individual movements given three different fuzzy-based models and compared it with an equivalent nonfuzzy, or crisp, model.

The dispersal model is described here and ecological interpretation of the results appears in Robinson and Graniero (2005a, b). In this model, the dispersal movement behaviour of an individual object consists of a movement decision and a residence decision. When an object is to move from its current location, it must decide on a destination location. Once at the new location, it assesses its surroundings by gathering information used in the residence decision. In other words, has the object found a suitable location, or does it continue to move on to another location? The fuzzy-decision model is one where relevant goals and constraints are expressed in terms of fuzzy sets. A decision is determined through an aggregation of the fuzzy sets (Bellman and Zadeh 1970, Klir and Yuan 1995). The goals and constraints for the movement and residence decisions are summarized in tables 2 and 3, respectively. In the case of both decisions, information about the surrounding landscape and conspecifics is of crucial importance. The spatial limits on this information are often referred to as the individual's perceptual range (Mech and Zollner 2002). In this model, the perceptual range is tantamount to being a spatial constraint on a query to the GIS database.

Table 2. Movement decision sets[a].

Equation	Description
$C^M = \Psi \cap F$	Constraint Set (C^M) constraining the search to those locations that are in the visible perceptual range (Ψ) and far from competing conspecifics (F).
$\Psi = P \cap L$	Visible Perceptual Range (Ψ), the degree to which a cell is both visible and falls within the perceptual range.
$P = \mu_p(x) = \begin{cases} 1 & \text{if} \quad d_x^c \le \beta \\ \theta(\beta - d_x^c) + 1 & \text{if} \quad \beta < d_x^c < \beta + 1/\theta \\ 0 & \text{if} \quad \beta + 1/\theta \le d_x^c \end{cases}$	The fuzzy set defining the 'ideal' perceptual range for a single individual. $X = \{x\}$ is a finite set of locations bounded by the limits of the study area. d_x^c is the Euclidean distance from the location of the dispersing animal object, c, to location x. The point at which $\mu_p = 1$ is represented by β and the parameter θ controls the rate at which $\mu_p \to 0$.
$L = \mu_L(x) = \max\left(\min\left(\frac{los_x^c - \alpha}{\beta - \alpha}, \frac{\gamma - los_x^c}{\gamma - \beta}\right), 0\right)$	The fuzzy set describing the degree to which location x is visible from a particular squirrel. The membership function for L is defined by a closed-form triangular function where los_x^c is the angle at which location x is visible from location c. If the local terrain creates a physical obstruction to visibility between c and x, then $L = 0$.
$F(x) = \mu_F(x) = 1.0 - \left(\bigcup_{k=1}^{c} \mu_{NC}^k(x)\right)$	The fuzzy membership of each location in the set of far_from_conspecific where if a conspecific is within the visible perceptual range (i.e. $k \in {}^{0+}\Psi$), then d_i^k is the distance from conspecific k to location i.
$NC^k(x;\ \alpha, \beta) = \mu_{NC}^k(x) = \begin{cases} \frac{\beta - d_x^k}{\beta - \alpha} & \alpha \le d_x^k \le \beta \\ 0 & \text{otherwise} \end{cases}$	The fuzzy set near_conspecific k where $\mu_{NC}^k(x)$ is the degree to which x is near conspecific k and d_x^k is the distance from conspecific k to x.
$G^M = A \cap \mathrm{I}$	Goal Set (G^M) degree to which a location is as near the edge of the perceptual range as possible and is forested.

Table 2 (*Continued*.)

Equation	Description
$A(x)=\mu_A(x)=\begin{cases}1 & \text{if forest}\\ 0 & \text{if non}-\text{forest}\end{cases}$	Habitat. In the case of this species that habitat would be forest. We use the crisp classification because it is unlikely, especially towards the edge of the perceptual range, that squirrels can evaluate vegetation in any detailed manner. Once an individual has moved to a location, then through exploratory movement, an evaluation of the habitat becomes more detailed.
$\mathrm{I}(x)=\mu_I(x)=\max\left(\min\left(1,\ \frac{d_x^c-\alpha}{\beta-\alpha}\right),0\right)$	Dispersal Imperative membership function where $\alpha=0$, and β is the distance of the farthest location in Ψ that has a non-zero membership value. Reflects the imperative of finding a home as far from the current location as possible, given the constraint of the perceptual range.
$D^M=C^M\cap G^M$	Decision set on first move, movement is to location with highest value. In case of ties, the first one in the list is chosen.
$B=\mu_B(x)=\left[\left(\left(\frac{\cos(q_p)+\cos(q(x))}{2}\right)^2+\left(\frac{\sin(q_p)+\sin(q(x))}{2}\right)^2\right)^{0.5}\right]^{\rho}$	The fuzzy set representing the degree to which a location falls within the set of direction_to_ move, where q_p is the direction, in radians, of the move to the current location and $q(x)$ the direction, in radians, from the current location (κ) to location x and exponent ρ functions like a *hedge*, we assume $\rho=2$.
$D^M=(C^M\cap G^M)\cap B$	Decision set on subsequent moves. Movement is to location with highest value. In case of ties, the first one in the list is chosen.

[a]This table is based on Robinson and Graniero (2005a), where the rationale for specific parameters and function forms is discussed in more detail.

In the movement-decision model, constraints consist of locations within the visible perceptual range and locations of conspecifics. The goal of an individual is to find a location as near the edge of the perceptual range as possible that is considered acceptable habitat and fits the set of constraints. Thus, the goal set is a function of the spatial arrangement of habitat and what we call dispersal imperative (table 2).

Once the object has moved to a location, it decides whether or not it is in a location suitable for stopping. In the residence-decision model (table 3), the object is constrained by whether or not its current location is sufficiently spatially separated from conspecifics that a home range can be established, while the goal is to have

Table 3. Residence decision sets[a].

Equation	Description
$C^R = \bigcap_c \mu^c_{Far}(\kappa)$	The Constraint Set (C^R) is a function of the spatial separation from surrounding conspecifics.
$\mu^c_{Far}(\kappa) = \begin{cases} 1.0 - \left\{ \frac{1.0}{\left[1 + \frac{d_c(\kappa) - \beta^c_{Far}}{\theta^c_{Far} - \beta^c_{Far}}\right]} \right\} & \text{if} \quad d_c(\kappa) \geq \beta^c_{Far} \\ 0.0 & \text{if} \quad d_c(\kappa) < \beta^c_{Far} \end{cases}$	The membership of location κ in the fuzzy set Far_from_conspecific c where $d_c(\kappa)$ is the distance from conspecific c (c=1 … k) and the current location (κ) of the Agent, β^c_{Far} represents the limit of a hypothetical core and θ^c_{Far} is the distance at which membership=0.5.
$G^R = LC \cap HA$	The degree to which location κ falls in the goal set G^R. In effect a measure of the degree to which the current animal location is habitat and contained within a large enough patch of habitat.
$\text{LC}(\kappa) = \mu_{\text{LC}}(\kappa) = \begin{cases} 1.0 & \text{if oak} \\ 0.9 & \text{if oak/deciduous\_bottomland} \\ 0.75 & \text{if deciduous} \\ 0.0 & \text{if conifer} \\ 0.0 & \text{if early\_successional\_deciduous} \\ 0.0 & \text{if wetland, pasture, grassland, ag.} \\ 0.0 & \text{if water} \end{cases}$	The degree to which a land cover type found in our GIS database can be considered quality habitat for a grey squirrel. The Agent uses the the land cover at the location, κ, where the squirrel has moved.
$\text{HA}(\kappa) = \mu_{\text{HA}}(\kappa) = \max\left(0, \min\left(1, \left[\frac{\text{farea}(\kappa) - \alpha_{\text{HA}}}{\beta_{\text{HA}} - \alpha_{\text{HA}}}\right]\right)\right)$	The degree to which location κ falls within the class of minimum habitat area. By setting the parameters α_{HA}=0.3 and β_{HA}=2.0 any patch less than 0.3 ha is clearly too small while any patch greater than 2 ha is clearly large enough.
$D^R = G^R \cap C^R$	The membership of location κ in the residence_location set
IF $D^R \geqslant 0.5$ THEN reside ELSE move	The decision rule for residence versus move.

[a]This table is based on Robinson and Graniero (2005a), where the rationale for specific parameters and function forms is discussed in more detail.

habitat of sufficient area. A decision rule is applied that leads to the Agent either taking up residence at the location or attempting another movement. The residence decision therefore functions primarily as a stopping rule.

The field of fuzzy sets and logic is exceptionally rich in methods for aggregation and combination. In this effort, the program SquirrelDispersal creates the World

object from layers drawn from a GIS database and activates the Agent classes of CompSquirrel, NoncompSquirrel, YagerSquirrel, and CrispSquirrel that correspond, respectively, to Agents using decision models based on compensatory, non-compensatory, Yager, and crisp aggregation methods. In addition, a nonfuzzy, or crisp, set version of the decision models was constructed. Each describes a particular class of Agents. For example, in table 2, the aggregation of sets to arrive at the decision set D^M for the first movement is expressed as a simple intersection of the goal and constraint sets such that

$$D^M = C^M \cap G^M \tag{4}$$

The non-compensatory aggregation is expressed as $D^M = \min(C^M, G^M)$. In contrast, the compensatory aggregation expressed as $D^M = (C^M \cdot G^M)$ allows for lower memberships in one set to be compensated to some degree by higher memberships in the other. The Yager method of aggregation is somewhat more complicated (see Robinson 2002). By setting the parameter p to 2, the Yager method defined the decision to move as

$$D^M = 1 - \min\left\{ \left[\left(1 - C^M\right)^p + \left(1 - G^M\right)^p \right]^{1/p}, 1 \right\} \tag{5}$$

Finally, the non-fuzzy, or crisp, set was defined as $D^M = ({}^{0.5}C^M \wedge {}^{0.5}G^M)$. Robinson and Graniero (2005a) provide a detailed listing of the aggregation methods used at each level of the decision process by each class of Agents.

Figure 8 shows a dispersal example of each type of Agent. The Agents were programmed with no Probe awareness of each other and do not modify the landscape in any way. Therefore, the agent-specific simulations were executed simultaneously on the same landscape with no interference. In general, the fuzzy Agents' pattern of behaviour was more plausibly realistic than that of the agents using crisp logic. This means that by using the ECO-COSM approach, a range of behaviours within a single species can be modelled. This is more realistic than assuming all agents have a single, identical decision model. In addition, the results shown in figure 8 illustrate how this approach yields plausible spatially explicit results for each individual agent and for the decision models in general.

Since squirrels tend to set up home ranges in areas not contested by conspecifics, we ran the simulations for landscapes with different levels of conspecific coverage. The baseline situation was the case of social fencing with no 'holes' in the conspecific landscape. Variations in the baseline were implemented by randomly allowing for 20, 40, and 60% of the landscape to be composed of unoccupied 'holes.' Figure 9 illustrates that variations in landscape perception, implemented via the ProbeWrapper, and consequent use of the information in the decision model, did have an effect on the Agents' ability to find a suitable location for establishing a home range. Figure 9 shows that Agents using fuzzy-decision models were consistently more able to find a suitable location for establishing a home range. This was especially evident in the situation where potential home range locations were in short supply (i.e. only 20% of all possible locations were available for use as a home range). This means that the Agents were able to navigate through the landscape in a more realistic manner to set up home ranges, whereas Agents using the non-fuzzy, crisp decision model were not as realistically successful. In fact, in the 20% case, none of the 'non-fuzzy' Agents were able to navigate the perceived

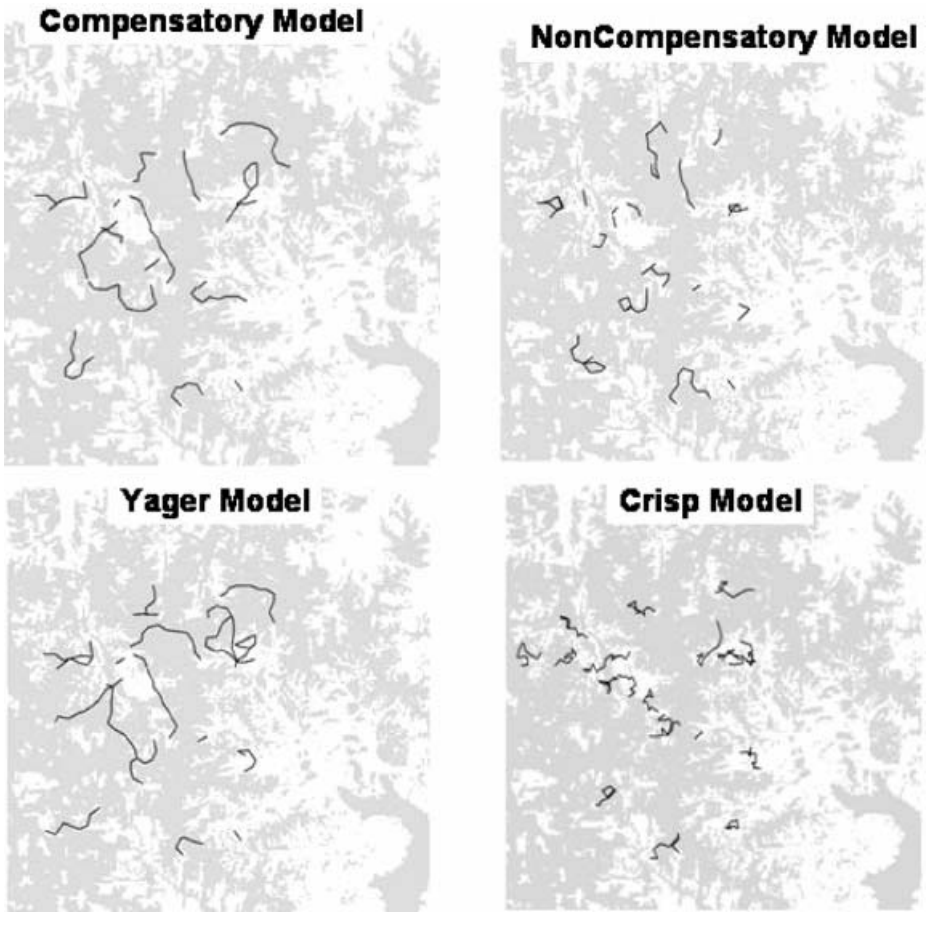

Figure 8. Example movement patterns for each agent decision model. The grey areas are least suitable as home range locations. The lines show dispersal paths for each squirrel agent. The same starting points were used for each decision model (from Robinson and Graniero 2005a).

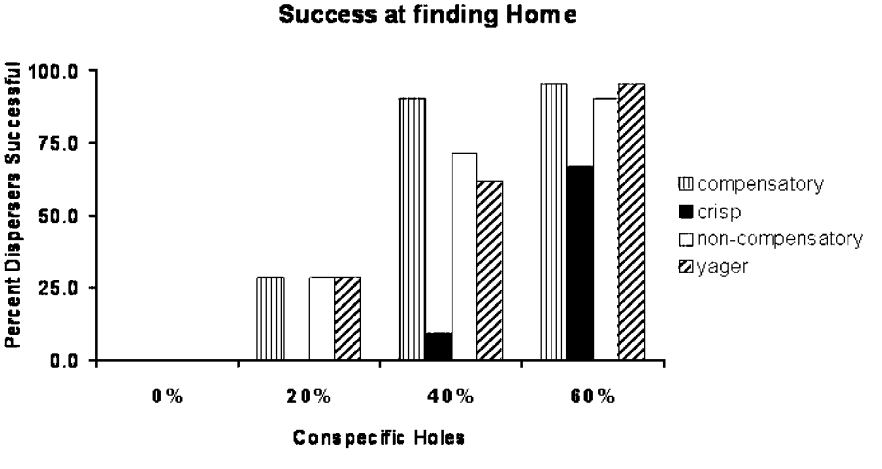

Figure 9. Percentage of squirrel dispersers in each decision model that found a location suitable as a home range.

landscape populated with conspecifics and find a location suitable for establishing a home range.

4. Adding agents to landscape models

ECO-COSM uses keyword-based identification of and access to World components, so an Agent only needs to specify which particular landscape layer is of interest at the conceptual level and does not need to be concerned about lower-level details such as the layer's spatial structure (e.g. vector vs. raster) or how the layer changes state as the model executes. The landscape layers could be statically stored in a GIS database as in the squirrel dispersal model, or they could be dynamically simulated as in the patch occupancy model. In both cases, the Agent's method of accessing, filtering, and using the landscape information is identical. It is possible to develop and test an agent-based model on a static landscape and then replace the static landscape layers with dynamically evolving landscape layers with no change to the Agent code, as long as the same keywords are used to name the layers. Probes provide a means to view the model state but not to change it; therefore, a Probeable Layer in a landscape model sees no difference in passing state information to a simple output file or to an active Agent. As such, an agent-based model and a landscape model may be independently developed and tested in ECO-COSM as stand-alone entities, and subsequently coupled by combining their initialization descriptions into a single setup routine. Further model refinement can allow Agents to register changes to local states in the landscape model's Layers, more closely coupling the models' interaction.

Nutrient patterns in wetlands are highly complex, and McClain *et al.* (2003) suggest that these patterns are driven by highly dynamic hotspot/hot moment biogeochemical processes in elemental cycling. A major barrier to improving our understanding of these patterns and processes is our ability to find these hotspots in the field with enough speed that we can collect sufficient data in their immediate surroundings for analysis and hypothesis testing. Movement and measurement decisions are based on current observations of the surrounding landscape, and the system is changing at a temporal scale similar to the field worker's. If such hotspot locations are not known a priori in the real world, what is the best search strategy to find them quickly in the field, especially if they can 'wink out'? Is it better to proceed with slow but complete local searching, or move faster with only a sub-sample of the adjacent region?

These questions can be explored with a dynamic landscape model generating realistic hotspot/hot moment dynamics, and agents simulating the field worker's decisions and movements. This section demonstrates the development of a simple landscape model of nutrient dynamics, and how field worker agents are added once the landscape model is developed. This example is clearly not sophisticated enough to properly answer the question posed above; it is simply intended to illustrate the role Probes can play in the landscape-agent model development process.

Consider a highly simplified system where a nutrient diffuses according to Fick's Law:

$$\frac{\partial u}{\partial t} = D\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \tag{6}$$

where u is the concentration of the nutrient, and D is the diffusion coefficient. The

Laplacian operator acts as a discrete, first-order approximation of Fick's Law on a grid with spacing h, and i, j representing row and column coordinates on the grid:

$$\nabla^2 u|_{i,j} = \frac{D}{h^2}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} + \right) + O\left(h^2\right) \tag{7}$$

The finite difference equation to estimate u at grid cell i, j at time $t+k$ is then

$$u_{i,j}^{t+k} = u_{i,j}^{t} + k\nabla^2 u|_{i,j} \tag{8}$$

where k is small enough to effectively ignore the truncation error $O(h^2)$. This finite difference equation is easily implemented as a StepRule operating on a *nutrient* Layer.

In most natural environments, the diffusion matrix is heterogeneous, often in a patchy pattern. Rather than applying a global diffusion coefficient D, the model uses a diffusion coefficient Layer, which is generated from Thiessen polygons surrounding random seed points i with a randomly generated coefficient D_i. A spatially variable depletion rate is applied to the region, with the edge cells having a rate 10 times higher than interior cells. This simply creates a 'drain' along the edge since the nutrient cannot continue diffusing beyond the extent of the modelled region. As an additional measure to reduce edge effects when identifying neighbour cells in the diffusion calculation, the model is constructed with a reflecting BoundaryTopology. Once a *source* Layer defining the local nutrient input rate is added, the result is a complete landscape model. At each time interval, the local state of the nutrient Layer is determined by neighbouring states and the local state at the corresponding location in the depletion and input Layers.

Rather than using a static input layer, consider that the nutrient appears in the system via asynchronous pulses at specific point locations. The *nutrient* Layer is removed, and nutrient inputs are modelled as Agents, parameterized with a location, input rate, time period over which the input source is 'on', and time period over which the input source is 'off'. Although this simple behaviour does not require the Agents to probe their environment, the Agent structure is convenient for having a number of discrete point objects with independent and similar, but not identical, behaviour and managing them as a Population.

A Population of seeker Agents is added to the model, each simulating a field worker with its own search strategy. On model initiation, each Agent obtains a Probe targeted at the *nutrient* Layer, analogous to the instrument the field worker would use to measure the nutrient concentration in the soil or water, as the case may be. In this example, the Probe produces perfect measurement results, though it would be easy to simulate documented instrument error using ProbeWrappers as described in section 2. The Agents are all placed at the same random location. After a delay period to let the landscape model reach stable conditions, the Agents begin to search for hotspots. Following some measure-and-compare search strategy, each Agent moves over the landscape until it determines that it must be at a hotspot and moves to a 'finished' Population. The simulation terminates once all Agents are in the 'finished' Population, or a set period of time has passed.

To demonstrate, two variants of a simple brute-force method are employed. To begin, the Agent measures the concentration at its current cell, then travels to each neighbouring cell (either the four cardinal cells of the von Neumann neighbourhood or all eight cells of the Moore neighbourhood) to take a measurement. If the original cell is the highest value, it is determined to be a hotspot; the Agent moves back to

that cell and decides it is finished. If it finds a neighbouring cell with a higher value, it moves to the neighbouring cell with the highest value, then starts the process again. The Agent acts with no memory; even if it just measured a cell, it still travels to it and re-measures. Once the Agent has determined the next cell it will visit, it schedules its next action to occur after the travel time it needs to reach the destination, where the travel time is simply the Euclidean distance between the source and destination cells. For example, if the Agent plans to move one cell to the east, it schedules its next action 1 time unit from now; if it plans to move to the south-east, it schedules its next action 1.414 time units from now. In the mean time, the landscape model independently follows its own scheduled activity, stepping through its calculations every k time units.

As a simple experiment, the model was executed 50 times with randomly established patterns of diffusion coefficients, hotspot locations, and hotspot behaviours: 10 diffusion patches with diffusion coefficient values between 0.35 and 0.65; and 40 point sources, with on- and off-intervals between 10 and 100 time steps. One 'von Neumann Seeker' and one 'Moore Seeker' were placed at a common random location, and they began seeking a hotspot after time 1000.

Figure 10 shows the result of an example model trial. The von Neumann Seeker found a hotspot in 40.11 steps, whereas the Moore Seeker found a hotspot in 50.89 steps. This was one of only two trials in which the Seekers found different hotspots. In general, the von Neumann Seeker found a hotspot faster than the Moore Seeker (table 4). However, the von Neumann Seeker found a hotspot in only 58% of the trials. In the other trials, the first exploration produced all zero values (i.e. no concentrations above background levels), giving no evidence of a hotspot. In

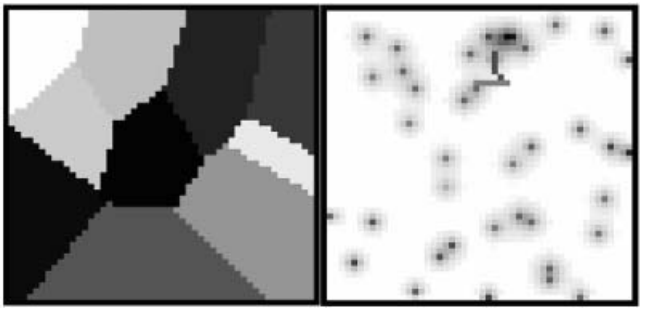

Figure 10. Example of a seeker model result. Left: patch pattern of the diffusion coefficient. Right: concentration map during final step, with seeker paths superimposed. Dark spots are hotspots. Dark grey path seeks in the Moore (eight-cell) neighbourhood; light grey path seeks in the von Neumann (four-cell) neighbourhood.

Table 4. General seeker performance statistics for 50 simulation trials.

Agent type	von Neumann Seeker	Moore Seeker
Success rate (agent finds a hotspot)	58%	82%
Average seek time, if successful	40.33	48.09
Fastest agent, when both find the same hotspot	74.1% of trials	25.9% of trials
Average seek time improvement over the other agent type (when fastest)	25.7%	5.1%

contrast, the Moore Seeker found a hotspot in 82% of the trials. Thus, if it is important to find a hotspot if it exists, this experiment suggests that it is better to opt for more careful, complete searching, even though it will usually (but not always) take longer to find a hotspot.

At this stage, the model is quite simplistic compared with the complex processes and goals operative in the natural world, but it clearly demonstrates the interaction among model components. The landscape model operates with three interacting grids: nutrient amount, diffusion coefficient, and nutrient-depletion rate. Two different types of modelled agents are operative: seeker Agents use Probes to examine state in the landscape model and therefore view their surroundings, make movement decisions, and carry out their motions; and nutrient input (hotspot) Agents change the local nutrient state through a controlled programming mechanism. Furthermore, the model components are operating in two different temporal modes. First, the landscape model, structured as an explicit finite difference model, calculates its steps on a regular, sub-unit timescale. Outputs of current model state are dumped via probes on a regular, unit timescale. Second, each Agent works on its own discrete event schedule, with intervals between scheduled activities determined by the time to traverse the landscape to the Agent's target destination.

The limit of complexity for an ECO-COSM model is defined by the computing platform rather than the framework design. Therefore, assuming the computing power is available, a modeller can construct a model of sufficient complexity to mimic the environment of interest and conduct an appropriate simulation experiment to explore and evaluate field-data collection and sampling design strategies.

5. Conclusions

ECO-COSM provides a simple modelling framework for constructing spatially explicit landscape simulations. It includes a Probe structure based on the Principle of Indirection and the *Delegation* design pattern that may be used to design complex data extraction and analysis experiments. This structure provides the fundamental elements to add agents to an existing landscape model. By using ProbeWrappers to modify the exact state returned from some location in a landscape Layer (or another Agent), sophisticated Agents can be constructed that:

1. use perceptual filters to modify their view of the landscape, or
2. make their movement decisions based on incomplete or uncertain observations.

A discrete-event scheduling mechanism allows simple, uniformly stepped landscape models to be executed simultaneously with discrete-event, trigger-oriented populations of agents. The modeller can develop dynamic landscapes with static observers, and static landscapes with dynamic observers. As each model is tested and refined in parallel, they may be merged together if they agree on a central conceptual model. This approach to coupled model development enhances the possibility of using 'virtual ecologists' to explore and understand ecosystems. The examples presented here serve as an illustration of the potential power of using ProbeWrappers to augment current agent-based modelling frameworks.

Acknowledgements

The authors gratefully acknowledge the financial support of individual Discovery Grants 44611–02 and 238431 from the Natural Sciences and Engineering Research Council of Canada (NSERC). We also appreciate the constructive efforts of the three reviewers which greatly improved the manuscript.

References

BANKS, J. and CARSON II, J.S., 1984, *Discrete-Event System Simulation* (Englewood Cliffs, NJ: Prentice-Hall).

BATTY, M., JIANG, B. and THURSTAIN-GOODWIN, M., 1998, *Local Movement: Agent-based Models of Pedestrian Flows*, CASA Working Papers, No. 4 (London: Centre for Advanced Spatial Analysis).

BELLMAN, R.E. and ZADEH, L.A., 1970, Decision-making in a fuzzy environment. *Management Science*, **17**, pp. 141–164.

BIAN, L., 2003, The representation of the environment in the context of individual-based modeling. *Ecological Modelling*, **159**, pp. 279–296.

BOOCH, G., 1994, *Object Oriented Analysis and Design with Applications*, 2nd edition (Reading, MA: Addison-Wesley).

BOOTH, G., 1997, Gecko: A continuous 2-D world for ecological modeling. *Artificial Life Journal*, **3**, pp. 147–163.

BROWN, D.G., RIOLO, R., ROBINSON, D.T., NORTH, M. and RAND, W., 2005, Spatial processes and data models: toward integration of agent-based models and GIS. *Journal of Geographical Systems*, **7**, pp. 25–47.

CASWELL, H. and COHEN, J.E., 1991, Disturbance, interspecific interaction and diversity in metapopulations. *Biological Journal of the Linnean Society*, **42**, pp. 193–218.

COLLIER, N., HOWE, T. and NORTH, M., 2003, Onward and upward: The transition to Repast 2.0. In *Proceedings of the First Annual North American Association for Computational Social and Organizational Science Conference*, Pittsburgh, PA, June 2003.

DEANGELIS, D.L., GROSS, L.J., WOLFF, W.F., FLEMING, M., NOTT, M.P. and COMISKEY, E.J., 2000, Individual-based models on the landscape: applications to the Everglades. In *Landscape Ecology: A Top-Down Approach*, J. Sanderson and L.D. Harris (Eds), pp. 199–211 (Boca Raton, FL: Lewis).

FORREST, S. and JONES, T., 1994, Modeling complex adaptive systems with Echo. In *Complex Systems: Mechanisms of Adaptation*, R.J. Stonier and X.H. Yu (Eds), pp. 3–21 (Amsterdam: IOS).

GAMMA, E., HELM, R., JOHNSON, R. and VLISSIDES, J., 1995, *Design Patterns: Elements of Reusable Object-Oriented Software* (Reading, MA: Addison-Wesley).

GARDNER, M., 1971, On cellular automata, self-reproduction, the Garden of Eden and the game 'life'. *Scientific American*, **224**, pp. 112–117.

GRANIERO, P.A., 2001, The effect of spatiotemporal sampling strategies and data acquisition accuracy on the characterization of dynamic ecological systems and their behaviors. PhD thesis, University of Toronto.

GRIMM, V. and RAILSBACK, S.F., 2005, *Individual-Based Modeling and Ecology* (Princeton, NJ: Princeton University Press).

GROSS, L.J. and DEANGELIS, D.L., 2001, Multimodeling: new approaches for linking ecological models. In *Predicting Species Occurences: Issues of Scale and Accuracy*, J.M. Scott, P.J. Heglund, M. Morrison, M. Raphael, J. Haufler and B. Wall (Eds) (Covello, CA: Island Press), pp. 467–474.

HOLLAND, J.H., 1992, *Adaptation in Natural and Artificial Systems*, 2nd edition (Cambridge, MA: MIT Press).

HOLLAND, J.H., 1994, Echoing emergence: Objectives, rough definitions, and speculations for Echo-class models. In *Complexity: Metaphors, Models and Reality*, G.A. Cowan, D. Pines and D. Meltzer (Eds), pp. 309–342 (Reading, MA: Addison-Wesley).

HOLT, R.D., PACALA, S.W., SMITH, T.W. and LIU, J., 1995, Linking contemporary vegetation models with spatially explicit animal population models. *Ecological Applications*, **5**, pp. 20–27.

HRABER, P.T. and MILNE, B.T., 1997, Community assembly in a model ecosystem. *Ecological Modelling*, **103**, pp. 267–285.

INCHIOSA, M.E. and PARKER, M.T., 2002, Overcoming design and development challenges in agent-based modeling using ASCAPE. *Proc. Nat. Academy of Science*, May 14, 99; Suppl. **3**, pp. 7304–7308.

JOHNSON, R. and FOOTE, B., 1988, Designing reusable classes. *Journal of Object-Oriented Programming*, **1**, pp. 22–35.

KLIR, G.J. and YUAN, B., 1995, *Fuzzy Sets and Fuzzy Logic: Theory and Applications* (Upper Saddle River, NJ: Prentice-Hall).

KRASNER, G.E. and POPE, S.T., 1988, A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, **1**, pp. 26–49.

KUKLA, R., KERRIDGE, J., WILLIS, A. and HINE, J., 2001, PEDFLOW: development of an autonomous agent model of pedestrian flow. *Transportation Research Record*, **1774**, pp. 11–17.

LUKE, S., CIOFFI-REVILLA, C., PANAIT, L., SULLIVAN, K. and BALAN, G., 2005, MASON: A multiagent simulation environment. *Simulation*, **81**, pp. 517–527.

MARTIN, R.C., 1996, The dependency inversion principle. *C++ Report*, **8**, pp. 1–12.

MARTIN, R.C., 2003, *UML for Java Programmers* (Upper Saddle River, NJ: Prentice Hall).

MCCLAIN, M.E., BOYER, E.W., DENT, C.L., GERGEL, S.E., GRIMM, N.B., GROFFMAN, P.M., HART, S.C., HARVEY, J.W., JOHNSTON, C.A., MAYORGA, E., MCDOWELL, W.H. and PINAY, G., 2003, Biogeochemical hot spots and hot moments at the interface of terrestrial and aquatic ecosystems. *Ecosystems*, **6**, pp. 301–312.

MECH, S.G. and ZOLLNER, P.A., 2002, Using body size to predict perceptual range. *Oikos*, **98**, pp. 47–52.

MINAR, N., BURKHART, R., LANGTON, C. and ASKENAZI, M., 1996, The Swarm simulation system: A toolkit for building multi-agent simulations. Available online at: http://xenia.media.mit.edu/~nelson/research/swarm/ (accessed 2 October 2005).

PARKER, M.T., 2001, What is Ascape and why should you care? *Journal of Artificial Societies and Social Simulation*, **4**, Available online at: http://jasss.soc.surrey.ac.uk/4/1/5.html (accessed 21 July 2006).

PARKER, D.C.,BERGER, T., and MANSON, S.M. (Eds) 2002, *Agent-based models of land-use and land-cover change: Report and review of an international workshop, October 4–7, 2001*, LUCC Report Series No. 6 (Bloomington, IN: Indiana University Press).

ROBINSON, V.B., 2002, Using fuzzy spatial relations to control movement behavior of mobile objects in spatially explicit ecological models. In *Applying Soft Computing in Defining Spatial Relations*, P. Matsakis and L.M. Sztandera (Eds), pp. 158–178 (Heidelberg: Physica-Verlag).

ROBINSON, V.B. and GRANIERO, P.A., 2005a, Spatially explicit individual-based ecological modeling with mobile fuzzy agents. In *Fuzzy Modeling with Spatial Information for Geographic Problems*, F. Petry, V.B. Robinson and M. Cobb (Eds), pp. 299–334 (Heidelberg: Springer).

ROBINSON, V.B. and GRANIERO, P.A., 2005b, An object-oriented approach to managing fuzziness in spatially explicit ecological models coupled to a geographic database. In *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*, Z. Ma (Ed), pp. 269–300 (Hershey, PA: Idea Publishing Group).

SCHMITZ, O.J. and BOOTH, G., 1996, Modeling food web complexity: the consequence of individual-based spatially explicit behavioural ecology on trophic interactions. *Evolutionary Ecology*, **11**, pp. 379–398.

SOUTH, A., 1999, Extrapolating from individual movement behavior to population spacing patterns in a ranging mammal. *Ecological Modelling*, **117**, pp. 343–360.

TOBIAS, R. and HOFMANN, C., 2004, Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, **7**, Available online at: http://jasss.soc.surrey.ac.uk/7/1/6.html (accessed 21 July 2006).

WEIDMANN, N.B. and GIRARDIN, L., 2005, Evaluating Java development kits for agent-based modeling. *Journal of Artificial Societies and Social Simulation*, **8**, Available online at: http://jasss.soc.surrey.ac.uk/8/2/8.html (accessed 21 July 2006).

WESTERVELT, J.D. and HOPKINS, L.B., 1995, Facilitating mobile objects within the context of simulated landscape processes. In *Third International Conference/Workshop on Integrating GIS and Environmental Modeling*, 21–25 January 1996, Santa Fe, New Mexico. Available online at: http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/sf_papers/westervelt_jim/paper.html (accessed 3 October 2005).

WESTERVELT, J.D. and HOPKINS, L.B., 1999, Modeling mobile individuals in dynamic landscapes. *International Journal of Geographical Information Science*, **13**, pp. 191–208.

WESTERVELT, J.D., 2002, Geographic information systems and agent-based modeling. In *Integrating Geographic Information Systems and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes*, H.R. Gimblett (Ed), pp. 83–104 (Oxford: Oxford University Press).

WOOD, J., 2002, *Java Programming for Spatial Sciences* (London: Taylor & Francis).