ELSEVIER

Ecological Modelling 141 (2001) 1–18

ECOLOGICAL MODELLING

www.elsevier.com/locate/ecolmodel

A domain-specific language for models of landscape dynamics

Andrew Fall *, Joseph Fall

School of Resource and Environmental Management, Simon Fraser University, Burnaby, BC Canada, V5A 1S6

Accepted 10 April 2001

Abstract

Gaining insight into the dynamic nature of landscapes often involves the use of simulation models to explore potential changes over long time frames and extensive spatial areas. However, bridging the gap between conceptual models of landscape dynamics and their simulation on a computer can lead to many pitfalls. If implemented using a general-purpose programming language, the underlying model becomes hidden in the details of the computer code, making it difficult to compare the conceptual and implemented models, and to modify the model. Alternatively, previously built models may contain hidden assumptions and have limited adaptability. *Domain-specific languages* have been developed in a number of areas to facilitate the construction of models at a level closer to the conceptual model, thereby making model implementation more accessible to domain experts. Such tools to support modelling in the domain of landscape ecology can achieve a balance between the flexibility of programming and the structure and ease of using pre-built models. One of the goals of SELES (Spatially Explicit Landscape Event Simulator) has been to create a language for modelling landscape dynamics that provides ecologists and planners with an appropriate tool to address some of the problems that arise in model development. Our high-level, structured language separates the specification of model behaviour from the mechanics of its implementation, freeing landscape modellers from programming and allowing them to focus on the underlying model. This language is declarative and thus permits a clear representation of the conceptual model, which the SELES engine converts into a computer simulation of landscape change. Our structured framework guides the development of a broad class of spatio-temporal landscape models by aiding model prototyping, modification, verification, comparison, and re-use.

Keywords: Cellular automata; Landscape ecology; Modelling languages; Spatio-temporal simulation

1. Introduction

Sustainable landscape management requires analyses that encompass large spatial scales, processes acting over long time frames, and heterogeneous units of study. The development of models

* Corresponding author. Tel.: +1-250-391-1918; fax: +1-604-2914968.

E-mail addresses: fall@cs.sfu.ca (A. Fall), jfall@sfu.ca (J. Fall).

0304-3800/01/$ - see front matter
PII: S0304-3800(01)00334-9

to complement empirical studies is critical for (i) extending our understanding of the natural processes driving landscape dynamics, (ii) predicting the consequences of management actions, and (iii) developing theory in landscape ecology (Baker, 1989; Turner, 1989; Sklar and Costanza, 1991). Spatially explicit landscape models that examine interactions over a range of temporal and spatial scales have been a key component in a number of studies (e.g. Gardner et al., 1989; Turner et al., 1989; Grant and French, 1990; Acevedo et al., 1996; Baltzer et al., 1998). The goals of such models have included predicting landscape-scale changes over time (e.g. Turner, 1988; Desmet and Govers, 1995), estimating the effect of disturbances on the spatial distribution of plant species (e.g. Moloney and Levin, 1996), assessing forest management plans (e.g. Wallin et al., 1994), and exploring assumptions in analytic models (e.g. Baker, 1992; Boychuk et al., 1997; Fall, 1998b; Lertzman et al., 1998).

To implement a conceptual, mathematical or statistical model as a computer simulation, it must be transformed into a set of computer instructions. Two common approaches are either to parameterize an existing model or to program the model directly. A fixed model with a predefined behaviour will likely be tailored to a specific purpose, scale, and ecological system. Thus, it possibly contains assumptions concealed within its implementation and may be difficult to adapt to new circumstances.

If a general-purpose programming language such as C++ is used, then the conceptual basis for a model may be overwhelmed by the details necessary to elucidate the desired behaviour. The majority of instructions in such programs are typically extraneous to the concepts in the underlying model — they control the flow of execution (e.g. loops), manage system resources (e.g. memory allocation), and manipulate data structures (e.g. sorting). These may obscure important semantic details of the model, leading to a number of consequences (for more discussion of these issues, see Derry, 1998; Lorek and Sonnenschein, 1998): (i) it is difficult to verify that that the program correctly implements the original model specification; (ii) the model will likely be hard to modify, and so its generality may be limited; (iii) computer models can usually only be compared by their inputs and outputs (Olde and Wassen, 1997); and (iv) integrating the model to work with other models or tools (e.g. GIS or visualization packages) can be difficult and is most often limited to the exchange of output files.

Domain-specific modelling systems address these difficulties by supporting the development of models within a particular area of inquiry. Such tools provide building blocks that match the user and the application (Barber, 1992) and are a fundamental characteristic of knowledge-based (or expert) systems (Merkuryeva and Merkuryev, 1994). We propose that languages specific to the domain of dynamic landscape modelling can help to overcome the problems associated with model implementation, to facilitate efficient construction of a wide range of models, and to open landscape modelling to a broader range of people. Our proposal is guided by the following principles:

1. Landscape ecologists and planners are not necessarily programmers — they should be able to transform their conceptual models into computer simulations without having to implement them in a general-purpose computer language.
2. Model behaviour and assumptions should be explicit. This is necessary to have confidence that the results of simulation experiments derive from the behaviour of the conceptual model and the landscape state, and not from implementation artefacts.
3. It should be possible to decompose complex models into semi-independent components to simplify conceptual model design, implementation, verification and experimentation.

SELES (Spatially Explicit Landscape Event Simulator) provides a structured framework to guide development and simulation of spatial landscape models. At the heart of SELES is a high-level, declarative modelling language used to specify processes acting in a model and a discrete-event simulation engine that interprets and executes such models. Our language is sufficiently general to specify a diverse range of model behaviours and types. Model specifications serve as a fairly clear description of their semantics, and

thus, models may be easily compared, modified, and reused.

The goal of this paper is to motivate the need for domain-specific languages that focus landscape model development at the conceptual level, and to present SELES as one such language. We first review techniques and tools available for constructing spatio-temporal models, and outline desirable goals and characteristics of domain-specific languages for landscape dynamics. Section 3 presents a conceptual view of the SELES modelling framework and the structure of our model specification language. We illustrate the utility of SELES in a difficult land-use planning problem — accounting for the unpredictable effects of fires in forest management. In Section 5, we evaluate the effectiveness of SELES as a domain-specific landscape modelling language.

2. Implementing dynamic spatial landscape models

The reviews by Baker (1989) and Sklar and Costanza (1991) both identify a class of models that embody explicit treatment of spatial relations and dependence (e.g. adjacency, flows), a large, heterogeneous spatial domain, dynamic feedback, and flows between elements (e.g. the role of processes in regulating or modifying landscape pattern, and vice versa). Techniques to design and implement such *dynamic spatial landscape models* are the focus of this section.

Modellers have employed a variety of approaches to untangle the beauty of a model from the beast of its implementation on a computer. These approaches may be viewed as lying on a spectrum of specificity as shown in Fig. 1, which also shows several example tools placed along this gradient. At one end lie general-purpose programming languages, which are very flexible and allow the widest possible range of models to be constructed. In addition to the problems described in Section 1, direct programming requires the largest investment in terms of time and skill, and a programming specialist may be needed to bridge the *modelling gap* (Lorek and Sonnenschein, 1999). At the other extreme lie packaged computer models that require only parameterization to run. If a suitable model could be found, it may provide the lowest cost tool, but users should be wary of the inherent limits to comprehension and adaptation of *black-box* models. In the middle of the spectrum are a variety of specialized tools that are designed to aid in the implementation of specific types and classes of models.

Program-level support tools extend the facilities available in general-purpose programming languages. Special-purpose simulation languages, like SimScript (CACI, 1987), aid implementation, but are generic tools and so share many of the disadvantages of programming. DEVS (Clark, 1992; de Vasconcelos and Zeigler, 1993) solves some of these problems by providing an object-oriented formalism for specifying dynamic, hierarchical, discrete-event simulations that facilitates structured model and program design. Another approach has been to create domain-specific systems via software libraries of useful objects (modules or object classes) for building specific classes of models. This 'programmer's toolbox' approach can

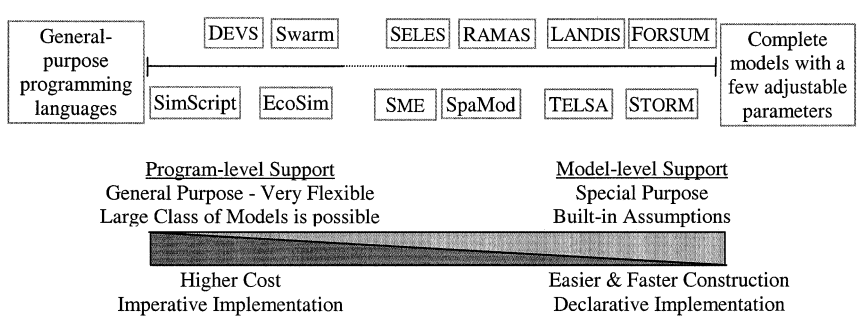

Fig. 1. Spectrum of approaches to implementing spatial landscape models.

substantially reduce implementation time and increase program reliability. Higher-level code, usually in a general-purpose programming language, specifies how the objects are used to produce the desired model behaviour. For example, EcoSim (Lorek and Sonnenschein, 1998) provides a set of C++ classes to assist construction of spatial landscape and individual-based models. Swarm (Minar et al., 1996) takes this approach one step further by providing a set of graphical tools for the display and analysis of model output. These approaches allow more focus on conceptual design, and so are well suited to modellers who possess good programming skills and require substantial flexibility.

Model-level support tools, however, assist model construction without requiring programming. A number of pre-programmed models have been developed that allow substantial flexibility for adaptation by providing an extensive set of parameters for which the values or definitions can be altered (e.g. FORSUM and STORM; Frelich and Lorimer, 1991; Krauchi, 1995). TELSA (Klenner et al., 1997) and LANDIS (Mladenoff et al., 1996) are more general examples of this approach. LANDIS represents the forest with a fixed structure that can be parameterised for an arbitrary number of tree species, each with unique ecological characteristics. TELSA allows both the number and type of spatial layers to be specified, as well as the number and complexity of relationships between layers. Both LANDIS and TELSA target modelling interactions between succession, natural disturbance and logging. These tools allow comparatively rapid model development, and provide a fairly straightforward mechanism for implementation. However, they also tend to be quite specific, and much of the model behaviour and assumptions are embedded in the program and may not be explicit or easily modified.

Towards the middle of the spectrum are domain-specific systems that provide model-level support for classes of models rather than individual model types. They make fewer assumptions about the underlying model structure than do pre-programmed models. For example, Petri nets (Groenwold and Sonnenschein, 1998) provide a graphical technique for describing cellular automata and have a formal semantics that enforces an unambiguous interpretation of the model specification. SPAMOD (Gao, 1996) automatically translates specifications of models based on partial differential equations or difference equations into C code that is then linked with a run-time environment. RAMAS-GIS (Boyce, 1996) supports exploration of spatial population dynamics in a static landscape, while WESP-TOOL (Lorek and Sonnenschein, 1999) aims to support conceptual individual-based modelling. PCRaster provides a declarative language for iterative environmental modelling that is integrated with a GIS (Wesseling et al., 1996). STELLA (Costanza et al., 1998) is a tool for graphically building and running aspatial stock and flow type models (Forrester, 1961). The Spatial Modelling Environment (SME) can make use of a STELLA model to specify within-cell dynamics in a spatial context (Maxwell and Costanza, 1997a,b). It additionally provides a declarative language for collaborative construction of modular spatio-temporal models, which are translated into C++ code and linked with the SME driver. SME and several other systems also focus on model inheritance in an object-oriented paradigm of modelling. Although it can assist in model specification, inheritance can also introduce assumptions, and so care must be exercised, particularly when inheriting from detailed or complex super-classes.

A characteristic of domain-specific tools is that they focus on a *framework* for specifying behaviour rather than a *parameterization* of pre-specified behaviour. A domain-specific language for landscape modelling provides model-level support by framing the implementation details in the context of landscape ecology. It should:

- be *simple* enough to make landscape modelling accessible to a wide audience. With a shallow learning curve, novice users should be able to write simple models and understand more complex models.
- be *capable* of supporting a wide variety of model types. Model complexity should be limited by knowledge and not system constraints. Constructing arbitrarily complex models should be possible through successive refinement.

- allow models to be composed as a set of autonomous interacting *components*, to facilitate formulation, verification, and experimentation.
- force assumptions to be *explicit* and behaviour to be *transparent*.
- be able to *efficiently* process relatively large models on commonly available hardware (e.g. desktop computers) or to transparently support execution on high-speed parallel processors.
- allow model *adaptation* to different circumstances to be relatively easy. Due to scale and landscape-specific issues as well as data availability, direct re-use is not always feasible.
- support *communication*, allowing non-modellers to be involved in the model-building process and providing an environment in which diverse sources of knowledge can be integrated.

The goals of simplicity, transparency and communicability imply that a language should be *declarative*, requiring only the salient model behaviour to be specified and letting the simulation engine fill in the details. By selecting a domain general enough to address a wide array of problems, yet specific enough to be contained, users can build reliable and richly expressive models to investigate complex problems in that domain. Developing a variety of domain-specific languages for spatial landscape research and management will be essential to provide ecologists, planners and others with a suite of tools to help resolve difficult ecological and socio-economic issues.

3. SELES landscape modelling framework

We consider landscape change to arise as the result of definable processes or entities that occur in specific places and at specific times. The behaviour of these processes is contingent on conditions (both prior and current), and their actions in turn change the state of the system, both locally and globally. In sum, these processes may be viewed as *agents of landscape change* that react to and modify the landscape state in various *spatio-temporal contexts*. In turn, a spatio-temporal context (or just context) is the set of information (i.e. values of the component variables) available at a particular time and place. Since contexts can exist at different spatial and temporal scales, and can be transient, they provide a general hierarchical framework (sensu O'Neill et al., 1986; Allen and Hoekstra, 1992) for describing landscape dynamics. In this sense, SELES is a language for creating a spatio-temporal state-space, defining behaviours to navigate through contexts in this state-space, and specifying state changes in those contexts. It can be used to study how consequences of changes in states (e.g. caused by disturbance agents, management regimes, etc.) combined with changes in structure and process (e.g. loss of connectivity between habitats, loss of species, etc.) interact to generate changes in ecosystem function. By managing contexts appropriately, users can create models of various forms, including cellular automata (Itami, 1994), individual-based simulation (Dunning et al. 1995), difference equations, discrete-event simulation (de Vasconcelos and Zeigler, 1993), and spatio-temporal Markov chains (Baltzer et al., 1998).

All SELES models are spatially explicit, with the spatial plane consisting of one or more raster layers (grids) of any arbitrary, but common, extent or resolution. The basic spatial unit is a cell. The SELES framework consists of our modelling language, a simulation engine to execute models, and a graphic modelling environment with a number of associated tools to help guide model development, run experiments, and display model results (Fall and Fall, 1999a,b). A SELES model consists of two components:

1. A set of raster layers and global variables together define the landscape state. These include layers that may change during the simulation (e.g. *CanopyDensity* and *ForestAge*), and layers that are static (e.g. *Slope* and *Elevation*). The initial state for each layer can be defined by a digital map from a GIS, a classified digital image from remote sensing, or a synthetic map from one of the SELES static landscape models.
2. A set of *landscape events* defines model behaviours and effects. A landscape event is a generic framework for describing the characteristics of an agent of change, and forms a semi-independent sub-model that usually reflects a single key process (e.g. seed dispersal,

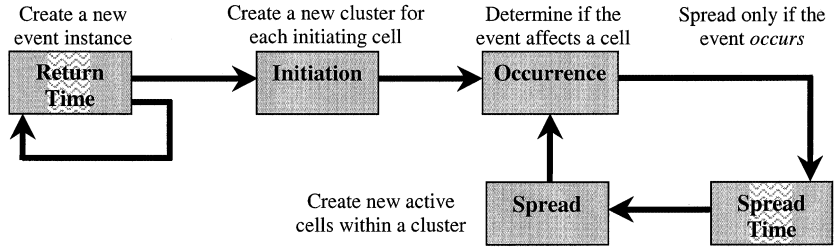

Fig. 2. Landscape events provide a general structure to model spatial processes. Using the event properties, modellers control the return of the event, the cells in which the event initiates, whether the event affects the cells in which it starts, and for spreading events, the rate of spreading and to where a cell may spread. The zigzag pattern indicates time intervals occurring between separate instances of an event, and during spread within the same event instance.

forest succession, ungulate grazing, timber harvesting, fire). A landscape event may have global characteristics (e.g. the interval between insect infestations on the landscape); local characteristics (e.g. the likelihood of infestations in forest stands of different ages); and a set of consequent effects or state changes (e.g. in cells where an insect infestation occurred, the layer *CanopyDensity* may be reduced and the layer *FuelLoad* may be increased). Feedback mechanisms between different processes are accomplished through state transitions on raster layers or global variables, with no direct inter-event communication.

A complete set of initial state information and landscape events forms a *simulation scenario*. A scenario can be processed by the simulation engine, either as a single run or as a set of Monte-Carlo runs. During each run, the SELES engine uses a priority queue to process events in their chronological order. Time in SELES is a continuous variable, to which the model builder ascribes the appropriate meaning (i.e. time steps may vary in size and processes with different temporal resolutions can be integrated). When a simulation begins, the specifications for all landscape events in the scenario are loaded into an internal format that facilitates efficient execution. Each such description forms a 'meta-event' that characterizes the general behaviour of all *instances* of this event (specific occurrences of the event on the landscape). The precise behaviour of an instance is influenced by both the event definition and the state of the landscape at the time it is processed.

Conceptually, landscape event instances *return* to the landscape, *initiate* in a set of cells, *occur* (i.e. establish) in some of these, and possibly *spread* to neighbouring cells (Fig. 2). Spreading is scheduled, and when processed selects a set of cells to spread to, continuing the loop shown. We distinguish between an event *initiating* and *occurring* in a cell to support modelling the development of an event within a cell (e.g. one may wish to distinguish between a fire starting in a cell and a crown fire that burns the entire cell). When an event initiates in a cell, it creates a *cluster* with one embedded *active cell*. A cluster in SELES is a general concept that links all active cells that derive from the same initiation point. New active cells are added to an existing cluster as an event spreads, and an active cell terminates if the event does not *occur* in it or after spreading. A cluster terminates when it no longer has any active cells, and an event instance terminates when it no longer has any clusters.

Modellers control this generic algorithm by specifying *properties* for an event (Section 3.2). Properties are the vehicle for creating active entities (i.e. event instances, clusters and active cells), and for moving through different spatio-temporal contexts. We provide a set of properties, each of which operates at particular spatial, temporal and organizational levels. Before we introduce these properties, we describe the conceptual basis for the state-space and contexts of an event.

3.1. Hierarchical state-space and spatio-temporal contexts

At any point during a simulation, there may be many event instances, clusters and active cells from the same or different events, and these active entities are organized hierarchically (Fig. 3). This dynamic hierarchy spans a range from the coarse-scaled resolution of global variables to the fine-scaled resolution of individual cells. Modellers have access to different levels in this hierarchy by declaring variables of the appropriate type (event, cluster or active cell), and these variables are automatically created and managed during model processing. Thus, modellers only need to understand the conceptual basis of this hierarchical structure and not how it is maintained. Our implementation of SELES takes advantage of efficient hierarchy management techniques (Fall, 1998a).

An important virtue of this hierarchy is that it permits communication among active entities at different levels of organization, allowing lower-order entities of the simulation (e.g. cell level) to act semi-independently but also to be co-ordinated by higher-order entities (e.g. cluster, event, and global levels). To illustrate, a cluster variable called *FireSize* may be declared as part of a *Fire* event to control the number of cells to which a particular *Fire* will spread. During initiation, the statement $FireSize = normal(\mu = 200,\ \sigma = 50)$ can be used to assign a random *FireSize* to each new cluster. The *FireSize* variable may be decremented by 1 each time a cell burns (during event occurrence), and the *Fire* can be extinguished when *FireSize* reaches zero. Thus, each *Fire* will spread until it burns *FireSize* cells. Such a model might also use an event variable called *AnnualAreaBurned* to track the area burned in a year and a cell variable called *Intensity* to track the magnitude of the fire in each burning cell.

The state-space of a landscape event thus includes the spatial and global variables that form the landscape state as well as variables associated with each active entity. During a simulation, the relevant values of spatial, cell, cluster and event variables are made available in the appropriate contexts. Global variables are available in all contexts. More precisely, we define a context in SELES as a particular subset of the variables in the state-space of an event and the values of those variables at a specific time, place and organizational level. A major goal of SELES is to provide a structured yet flexible tool with which contexts can be managed in order to produce the desired dynamics of a model.

By controlling the general algorithm shown in Fig. 2, modellers can build landscape events that

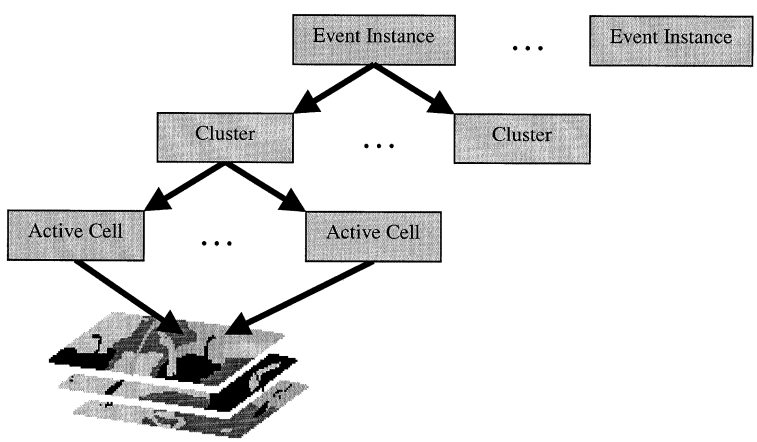

Fig. 3. SELES manages a hierarchical state for active entities. Each return of an event creates an event instance. Initiation creates clusters, each with one active cell. Spreading creates additional active cells in the same cluster. Variables declared at each of these levels are automatically created and managed during processing.

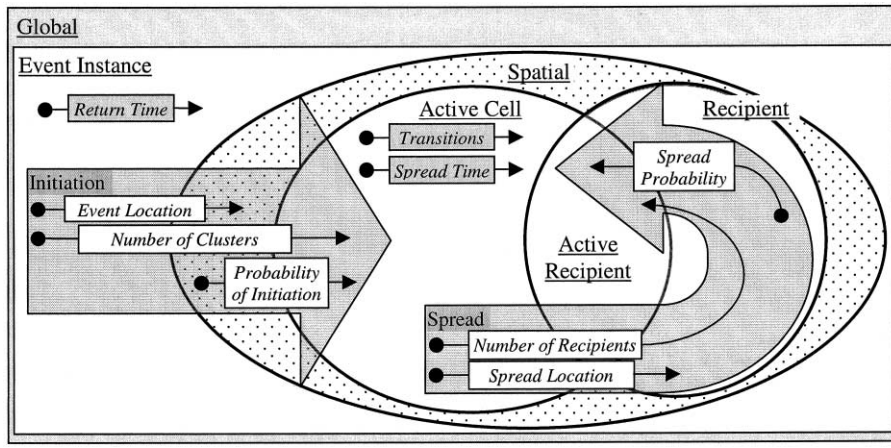

Fig. 4. Each property operates in a specific spatio-temporal context (indicated by ●), which is modified by its main expression (indicated by ►). There are six classes of contexts: global, event instance, spatial (i.e. at a specific location), active cell, recipient (i.e. at a spatial location reached during spread), and active recipient (i.e. in an active cell created during spread). *Return Time* moves to a new event instance context (at a later time) from an existing one. Initiation enters a set of active cell contexts from an event instance context. Event occurrence (*Transitions*) and *Spread Time* both operate within active cell contexts. Spreading enters a set of active recipient contexts from an existing active cell context. After spreading, the active cell terminates, transforming these active recipient contexts into active cell contexts.

enter approporiate contexts, and react to and modify the state within those contexts. All contexts exist at a specific time, but may differ in spatial scale and dynamic level. We define the following classes of contexts:

- *Global* contexts provide access to global information in an aspatial situation (i.e. where there is no specific location or active entity).
- *Event instance* contexts provide access to both the global information and the variables associated with an event instance (also in an aspatial situation).
- *Spatial* contexts have a specific location and so provide access to the values in raster layers at that location. In this paper, we only consider spatial contexts that have been accessed by an event instance, and thus a spatial context is a special type of event instance context.
- *Active cell* contexts are spatial contexts that are associated with an active cell. These contexts provide access to all the information in the above contexts as well as the variables associated with a cluster and an active cell.
- *Recipient* contexts occur during spread but *before* cells are actually selected for spread. These are spatial contexts in which there is no active cell, but they are associated with an active spreading cell. Thus, they provide access to the information available in a spatial context in addition to the information available in the active cell context of the spreading cell. Recipient contexts are important to provide a means of information flow during spread.
- *Active recipient* contexts combine active cell and recipient contexts and occur during spread *after* cells are selected for spread. These contexts are thus associated with an active cell as well as an active spreading cell and hence provide access to the information available in a both active cell contexts and recipient contexts.

The relationship among types of contexts is shown in Fig. 4. In this diagram, the enclosed areas represent sets of contexts (i.e. portions of the state-space) that may be accessed during a simulation. The most specific contexts are active recipient, which form the intersection of active cell and recipient contexts. These are both special cases of spatial contexts, which in turn are a subset of event instance contexts. One may alternatively view a context type as representing the set of variables available within contexts of that type. In this case, Fig. 4 shows the complements

of these sets (i.e. global contexts have the smallest set of available variables, and active recipient contexts have the largest set).

Table 1
Landscape event properties, each of which controls a specific aspect of an event's behaviour

Property name	Purpose
Return time	Defines the interval of time between successive instances of the event on the landscape. This may be a fixed or variable time-step.
Event location	Defines the set of cells in which the event can potentially initiate. This property can be used to restrict an event to a particular spatial region or landscape element.
Number of clusters	Defines the number of cells in which the event will initiate. This property works in conjunction with Probability of Initiation.
Probability of initiation	Defines the relative or absolute probability that the event will initiate in a particular cell, allowing landscape pattern to affect the spatial distribution of the process.
Transitions	Defines whether the event occurs in a cell, differentiating between initiation and establishment of an event in a cell. Spreading will only ensue if the event occurs.
Spread time	Used in spreading events to define the interval of time required for an event to spread from the current cell to its neighbours.
Spread location	Defines the set of cells to which an event can potentially spread from a cell. This property is analogous to Event Location except that spreading has a source cell.
Number of spread recipients	Defines the number of cells to which the event will spread from an affected cell. Analogous to Number of Clusters, except that spreading has a source cell.
Probability of spread	Defines the absolute or relative probability that the event will spread to a particular cell. Analogous to Probability of Initiation, except that spreading has a source cell.

3.2. Navigating through contexts via landscape event properties

The specific behaviour of a landscape event is determined by a set of *properties* (Table 1), each of which defines some characteristic of the underlying process being modelled. All properties are optional, so events can be succinctly specified. Each property is processed in a particular context called the *operating context*. When a property is evaluated, it may create active entities in the hierarchical state-space of an event, transform or filter the operating context, or enter new contexts. These resulting contexts are collectively called the *consequent contexts*. Modellers can specify behavioural dependencies and state changes in both types of contexts associated with a property. Understanding these mechanisms is the key to successful modelling in SELES, but comprehensive coverage for each property is beyond the scope of this paper (see Fall and Fall, 1999b). Instead, we focus on the motivation for the different properties and describe their roles in the general algorithm described above.

The *Return Time* property specifies the time of the first event instance (at simulation start-up) and the interval of time between subsequent instances. Processing an event instance involves *initiation* followed by evaluation of *Return Time* to create and schedule a new instance representing the next occurrence of the event. During initiation, the *Event Location* property determines the potential spatial locations (i.e. cells) in which the event might start, entering a set of spatial contexts. These may consist of the whole map, a particular spatial region, or cells that meet some criteria based on their state. The *Probability of Initiation* and *Number of Clusters* properties are used in conjunction to select a subset of these locations in which the event will actually initiate. This will create a set of clusters within the event instance, each with one embedded active cell, and transform the corresponding spatial contexts into active cell contexts. Fig. 5 illustrates how various contexts are entered and transformed during this process. Different uses of the above three properties can achieve a variety of behaviours, from initiation in every cell (e.g. for a *Succession* event),

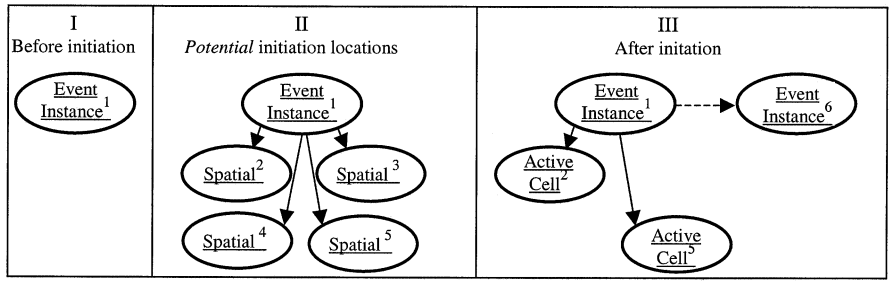

Fig. 5. Initiation creates a set of active cell contexts (for the initiating cells) from an event instance context. Superscripts indicate the identity of different contexts that are selected and transformed during this process. An instance is processed, starting in its associated event instance context (I). During initiation, the potential initiating cells have spatial contexts but are not active (II). A new active cell is created for each cell selected to initiate the event (III). The event instance remains active as long as it has at least one active cell. After initiation, a new event instance is created and scheduled for processing according to *Return Time*.

in a fixed number of cells (e.g. for a *Logging* event), or in a stochastic or emergent number of cells (e.g. for a *Wildfire* event).

The *Transitions* property defines whether an event *occurs* in a cell once it has initiated or spread there and operates within an active cell context. The semantics of 'occur' varies widely among events and is usually defined by a set of state changes associated with this property. If the event can spread, then the *Transitions* property also determines if the event should spread from the current cell.

Spreading is analogous to event initiation, except that there is an active spreading cell. The *Spread Time* property specifies when the event should spread from an active cell, and operates within its active cell context. When spreading is processed, the *Spread Location* property determines the possible spatial locations to which the active cell might spread, entering a set of recipient contexts. By default, this region consists of the cell's four cardinal neighbours, although one may specify an arbitrary neighbourhood or region. The *Probability of Spread* and *Number of Spread Recipients* properties are evaluated to select a subset of these locations to which the event will actually spread. This will create new active cells within the current cluster and transform their corresponding recipient contexts into active recipient contexts. During spread, the context of the spreading cell is available for potential and actual recipients to access, which is important for many models. After spreading has completed the spreading cell terminates, and so these active recipient contexts become active cell contexts. The *Transitions* property is then evaluated for each new active cell, and this may in turn cause more spreading via the event loop (Fig. 2). Fig. 6 illustrates how various contexts are traversed during spread.

Fig. 4 elaborates on the general behaviour of landscape events from Fig. 2, and shows how properties enter or transform their consequent context (indicated by ►) from their operating context (indicated by ●). For example, during initiation, *Event Location* starts in an event instance context and enters a spatial context for each location in the specified region. *Number of Clusters* enters an active cell context for each initiating cell from its event instance context. *Probability of Initiation* filters the spatial contexts selected by *Event Location*, transforming those associated with the initiating cells to active cell contexts. To illustrate, a *Succession* event might use the *Event Location* property to restrict processing to the forested cells of a landscape. The *Probability of Initiation* property will be evaluated in each of these spatial contexts, and may select a subset that can potentially modify a layer called *Species*. Each of these active cell contexts forms an operating context for the *Transitions* property, which may select the cells in which a change to the *Species* layer actually occurs.

Note that landscape event properties specify only the behavioural characteristics of an event

and not the actual state changes caused by the event. State-change specifications can be *associated* with the operating and consequent contexts of properties, and can access or modify variables available in those contexts. Since each property provides access to different contexts, modellers have flexibility as to where and when state changes can be made. The separation of model behaviour from effect makes events highly adaptable and allows modellers to specify state changes at appropriate scales. The ability to selectively choose properties and to control how they interact with the state allows modellers to describe a wide range of processes with one generic structure.

3.3. Specifying landscape events

Landscape event specifications consist of two components: (i) a definition of the hierarchical state-space for the event, and (ii) the properties and associated state changes that define the behaviour and effects of the modelled process. Here, we focus on the structure of the language rather than on its syntax. Simply declaring appropriate variables of each type (layer, event, etc.) specifies the state-space. Properties are specified with three parts: *Preliminary state-changes*, *Main expression*, and *Consequent state-changes*. The *main expression* defines the value of the property, which drives its behaviour (Table 1). State changes are specified as assignments of the form *variable = expression*. *Preliminary state-changes* are often used to initialize part of the state-space for the main expression, and are evaluated immediately before the main expression in the operating context of the property. Significant effects are generally made with *consequent state-changes*, which are evaluated in the consequent context(s).

Functions and distributions are used to construct expressions, and form the building blocks of the SELES modelling language, in an analogous way that functions are building blocks in spreadsheet programs. Expressions can range from simple constant values to complex compositions of sub-expressions. Many of the functions available are standard parts of numerous modelling systems, such as constants, continuous functions (e.g. linear, exponential, and trigonometric), relations (e.g. comparisons such as *less than* and *between*; logical connectives such as *and* and *or*), and conditional expressions (e.g. if–then–else). Some of the expressions that are more specific to landscape dynamics modelling and SELES include distributions (e.g. normal, Weibull, negative exponential), spatial functions (e.g. selecting a region of cells, summing a sub-function over the neighbourhood of a cell, or determining distances and directions), and classified functions that are used for categorical data, which is common for spatial information (e.g. community type or species).

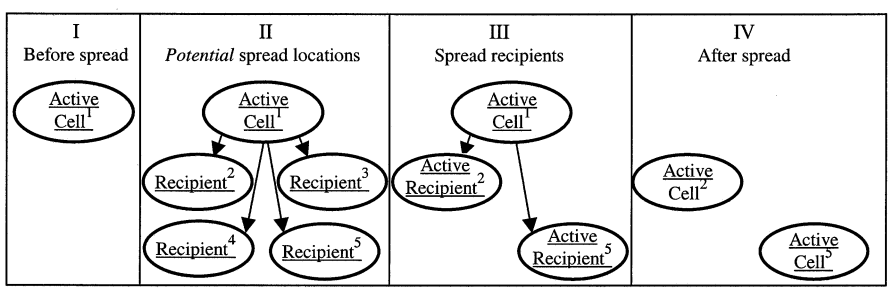

Fig. 6. Spreading creates a set of active cell contexts (for the recipient cells) from the active cell context of a spreading cell. Superscripts indicate the identity of different contexts that are selected and transformed during this process. Spreading starts from the context of an active cell (I). During spread, the potential recipient cells have spatial locations and have access to the active cell context of the spreading cell, but are not themselves active (II). A new active cell, in the same cluster as that of the spreading cell, is created for each recipient selected resulting in active recipient contexts (III). An active cell terminates after spreading, transforming the contexts of the recipients to active cell contexts in (IV).

Table 2
Adapted mean fire return interval (in years), and average and range of fire sizes (in hectares) by NDT zone for the zones in the Iskut-Stikine LRMP area

NDT zone	Average return (years)	Average fire size (ha)	Fire size range (ha)
1	150–700	50–500	0.1–10 000
2	150–350	150–200	0.1 to >25 000
3	75–200	50–10 000	0.1–200 000

4. Case study: modelling fire in a large landscape in northwestern B.C.

We intend to use the following case study to illustrate the use of SELES in a real application. It is outside the scope of this paper to provide sufficient details to convince the reader of its validity or utility. Rather, our goal is to demonstrate how SELES can facilitate the step from a conceptual model to its implementation in a relatively complex study that combines models of both management and natural processes. In the model description, property names are in italics, event names are in bold italics, and variable names and constants are in underlined italics. Segments of SELES language are shown in typewriter font.

A number of decision-making processes have been initiated in the province of British Columbia, Canada to address the need to balance ecological, social and economic interests in landscape management. One such process, called a Land and Resource Management Plan (LRMP), has been initiated in the Iskut-Stikine region in northwestern B.C., covering an area of approximately 5 million hectares. In collaboration with researchers in the provincial forest service, we constructed a SELES scenario to explore the interactions between wildfire and timber harvesting. Our goals were to illustrate the dynamic nature of the landscape to the LRMP participants and to provide a tool to explore the long-term effects of planning decisions.

Our main focus was to assess likely changes in forest age and stand species composition (e.g. dominant and secondary tree species) across the landscape. We developed sub-models (landscape events) for forest succession, fires and logging, using a spatial resolution (cell size) of 25 ha. In a series of short workshops, we met with experts on local ecology, natural disturbance and management to build the specifications for each sub-model. We based the succession model on a literature review and expert advice, and it responds to landform, stand-initiating disturbances, and biogeoclimatic zone. We based the logging model on discussions with a local planner, and it is primarily driven by the volume of wood to be harvested annually. This stochastic model produces harvest patterns consistent with recent harvesting activity, where the location of individual cut-blocks is controlled by a number of overlapping spatial and non-spatial constraints, including access constraints (i.e. a stand must have road access, where a *DistanceFromRoad* layer is updated by the harvest event) and adjacency constraints (e.g. a stand cannot be harvested adjacent to a stand under 15 years old).

We focus the rest of our discussion of this case study on the fire model. Our fire model combines statistical and empirical models of fire frequency and size with a more process-based model of ignition and spread. We combined empirical data from historic fires with a statistical component derived from a provincial-scale analysis by the B.C. forest service of fire return intervals and sizes by 'natural disturbance type' (NDT), which are zones with similar fire regimes (Province of British Columbia, 1995). Components of local fire behaviour were taken from expert opinions. Table 2 shows the mean fire return interval (MFRI) and size distribution by NDT.

Broad analysis of the historical fire return data reveals that total spring precipitation has a large influence on annual fire behaviour, and years can be classified as wet, dry or very dry with a distri-

Table 3
Expected size of individual fires (ha), and expected fire density expressed as the number of fires/100 km^2 (in parentheses) according to NDT zone and spring precipitation type

NDT	Wet	Dry	Very dry
1	$10^2 = 100$ (0.03)	$10^3 = 1000$ (0.06)	$10^{3.5} = 3162$ (0.06)
2	$10^2 = 100$ (0.025)	$10^{3.5} = 3162$ (0.05)	$10^4 = 10\,000$ (0.05)
3	$10^2 = 100$ (0.02)	$10^4 = 10\,000$ (0.04)	$10^{4.5} = 31\,623$ (0.04)

bution derived from the fire history study. We estimated the expected fire size and annual number of fires per 100 km^2, according to the spring precipitation type, to arrive at a MFRI that is consistent with Table 2. We computed return intervals based on these expected values according to NDT zone and spring precipitation type, where drier years result in an increase in average fire sizes and numbers as shown in Table 3. Combining the expected number and size of fires, and proportions of spring precipitation types, we obtain the expected area and percentage burned annually, and the MFRI for each NDT zone and over the entire landscape as shown in Table 4. According to this table, there is an approximately 200 year MFRI over the landscape using this model. The effects of suppression are difficult to assess from the historical database. We do not cover this aspect of our model in the case study.

4.1. Fire model specification

We focus here on key aspects of the fire model specification to illustrate how we used the SELES language to concisely specify model behaviour. To emphasize how SELES helps speed model development, this fire model only required a couple of days to implement and just over 100 lines of language syntax, in addition to the state-space declarations. Implementation time was overshadowed by the time required for background research, conceptual model development, data preparation and analysis, simulation experiments and output analysis, and documentation. Thus, as should be the case, model development effort was focused on model formulation and communication.

4.1.1. State-space definitions

Spatial layers include the cover type (*Landform*), forest age (*AgeInYears*), leading and secondary species (*Inventory*), NDT zones, and distance to nearest road. A portion of the declaration for *Landform* is shown below. The 'feature' identifiers are optional, but allow a class name (e.g. *activeFloodplain*) to be used in place of the class number in the model:

```
LAYER: Landform
   FEATURE 1: activeFloodplain
   FEATURE 5: spruceBog
   FEATURE 9: generalForest
   FEATURE 12: wetland
```

Global variables include an output variable to store the area of forest burned each year (*AnnualAreaBurned*). A number of hierarchical state variables were also required. The event variable *YearType* holds the value of the current fire year type (wet, dry or very dry), while *CurrNDT* holds the value of the current natural disturbance type for each event. A cluster variable *OpeningExtent* is used to hold the number of cells a particular

Table 4
Expected area and percentage burned/year and return interval by NDT zone

NDT	Expected area burned (ha)	Expected percentage burned annually	Mean fire return interval (years)
1	703	0.15%	654
2	3672	0.34%	290
3	5527	0.87%	115
Total	9902	0.46%	218

fire has left to burn. Below are the declarations for three of these:

```
GLOBAL VARIABLE: AnnualAreaBurned
EVENT VARIABLE: YearType
CLUSTER VARIABLE: OpeningExtent
```

4.1.1.1. Fire return time. We model fire on a yearly time-step, such that all fires that burn in a single year are processed simultaneously. The following shows the specification for the *Return Time* property for the *Fire* event. In general, properties are identified by their name, and are enclosed with tags if there are associated state changes. The first expression below sets the value of the property to 365.25 (using a day as the time unit). The consequent state-changes for this property are evaluated each time an event instance is processed, and will select a value (*Wet*, *Dry*, or *VeryDry*) for *YearType* according to the probabilities in the discrete distribution.

```
RETURNTIME
   RETURNTIME = 365.25

   /* Select a spring precipitation type */
   YearType = CLASSIFIED_DIST
                Wet: 0.83
                Dry: 0.15
                VeryDry: 0.02
              ENDFN
   /* Reset area burned annually */
   AnnualAreaBurned = 0
ENDRT
```

4.1.1.2. Fire initiation. During each year, the expected number of fires is determined for each NDT according to the current year's spring precipitation type and the size of the NDT zone using a table lookup (Table 3), and this is stored in the local variable *ExpectedNumFires*. Expected fire size is determined similarly, as a table lookup (also Table 3) to determine the fire size exponent, which is stored in the local variable *LogExpectedFireSize*. We select the actual number of fires for the year, and the actual size for individual fire patches, stochastically from negative exponential distributions. This is consistent with the fire history study and introduces a measure of variability into the model to reflect the unpredictability of these processes. The size of each fire is stored in the cluster variable *OpeningExtent*. The following segment of the *Number of Clusters* property shows how part of this behaviour is specified. The main expression causes a random number of clusters to be created, each of which has a unique *OpeningExtent* variable. The subsequent assignment is evaluated once for each cluster. The distribution NEGEXP returns a value drawn from a negative exponential distribution, and ' ∧ ' indicates exponentiation.

```
  NUMCLUSTERS = NEGEXP(ExpectedNumFires)
   ⋮
  /* Divide by CellSize to convert from hectares to cells */
 OpeningExtent = (1/CellSize) * NEGEXP(10^LogExpectedFireSize)
ENDNC
```

The locations at which to start the fire clusters are chosen stochastically from the forested portion of the landscape using the *Probability of Initiation* property, which incorporates human-caused fires ignited by increasing the chance of ignition in areas less than 5 km from a road. Thus, during each year, the model starts the chosen number of fires for each NDT zone, pre-selecting a size to burn for each fire.

4.1.1.3. Fire effect. Since the modelled fires are stand replacing, the effect of burning is to set the stand age to zero and the inventory type to 'burned'. The succession sub-model takes care of determining the next dominant species for the cell, based on the inventory prior to the fire. The *Transitions* property follows:

```
TRANSITIONS
   /* Make a transition only if there is extent to burn */
   /* AND if the stand didn't already burn this year */
   TRANSITIONS = (OpeningExtent > 0) AND (AgeInYears > 0)

   /* Primary state changes apply to active cells in these */
   /* three spatial layers */
   AgeInYears = ZERO
   PreviousInventory = Inventory
   Inventory = Burned

   /* Decrement the # of cells to burn for this cluster */
   OpeningExtent = OpeningExtent - 1

   /* Increment the area burned, where CellSize is a */
   /* constant defined as 25ha, to make the model portable */
   AnnualAreaBurned = AnnualAreaBurned + CellSize
ENDTR
```

4.1.1.4. Fire spread. Since fires are modelled annually, the rate of spread is instantaneous. Fires are assumed to only be able to spread to forested cardinal neighbours (up, down, left and right), the default *SpreadLocation*. Spreading will continue until the pre-determined fire size has been reached (i.e. *Transitions* evaluates to false). Once all the fire clusters have finished burning, the *AnnualAreaBurned* is written to the output database. The specification for *Probability of Spread*, shown below, ensures that fire spreads only to burnable cells since the expression will evaluate to TRUE (1) only if the *Landform* in the cell has a forested value. More complex versions of spreading take into account the effects of elevation and aspect (i.e. fires tend to spread uphill), differential susceptibility of dominant species, and fire 'spotting' to non-adjacent cells.

```
/* Only spread to forested cells. Evaluates to 0 (FALSE) or */
/* 1 (TRUE) */
SPREADPROB = (activeFloodplain <= Landform <= generalForest)
```

5. Discussion and conclusion

Regardless of the complexity of a conceptual model of landscape dynamics, implementation poses a number of challenges. Domain-specific languages are model-level support tools that provide high-level languages for specifying model characteristics. They support declarative specification at the semantic level rather than a procedural or step-by-step specification (Maxwell and Costanza, 1997b). In order to have utility, such systems must encompass a domain that can address a broad range of problems with enough constraints to be conceptually tractable. We believe that the domain of landscape ecology and management warrants specific languages and systems.

SELES is a system that supports a particular view of landscape dynamics expressed through the semantics of landscape events. Through the concepts of *hierarchical active entities*, *spatio-temporal contexts*, and *event properties*, SELES provides a general language for creating models to explore landscape ecology and management concepts and issues. The class of dynamic models that can be built with SELES is necessarily restricted, but our framework provides guidance for implementing models within this class. It addresses the desirable characteristics of a domain-specific language outlined in Section 2 as follows:

- *Simplicity*: Constructing model prototypes often takes a few minutes to a few hours. Models can be enhanced through successive refinement. Although complex models may be difficult for novice users to construct, the coherency of the framework allows such models to be comprehended. The Windows NT version has a well-developed user interface to simplify setting up and running simulations.
- *Flexibility*: We have constructed models of natural disturbance, management and succession, as well as habitat suitability, meta-population, and connectivity models. Behaviour types have ranged from periodic to episodic and empirical to process-based. The restrictions imposed by our framework provide guidance for model development. Potential enhancements include adding new properties to increase control of context access and navigation.
- *Capability*: SELES models do not have specific data and model requirements. Users include the level of detail suitable for the modelling task. This is an advantage shared with program-level support tools, but few other model-level support tools. The primary restrictions in SELES are that spatial information must be raster-based and that models must be discrete-event simulations.
- *Modularity*: A scenario is decomposed into a set of semi-independent sub-models represented by landscape events. Landscape events are further decomposed into properties, each responsible for key components of model behaviour. This modularity eases the conceptual burden of modelling complex systems, while using the same structure for all landscape events and properties reduces the learning curve for the system. Landscape events can be selectively included in a scenario without modifying the other events, facilitating verification, refinement, and experimentation.

- *Transparency*: Our language ensures that model specifications are transparent and that all specific behaviour is explicit. The only assumptions that users must be aware of are the semantics of landscape events.
- *Efficiency*: Although landscape events are not currently compiled, their structure is optimized when they are loaded, and they are processed using optimized algorithms. On an average desktop computer (200 MHz Pentium with 128 Mb RAM), we have run scenarios with 20–30 spatial layers for landscapes with hundreds of thousands of cells. The Iskut-Stikine model has over 250 000 cells and takes several seconds per simulated year. The efficiency of a programmed model can vary dramatically and depends on the level of effort invested in implementation. Many models have a relatively short life-cycle that does not warrant the cost associated with optimization, so the potential efficiency benefits of programming a model may not be realized. However, optimization is justified for domain-specific systems since they have longer development cycles than individual models.
- *Re-usability and adaptability*: Due to the transparent, modular nature of SELES models, adapting landscape events to other modelling projects is straightforward. In general, some aspects of any model will be problem- and region-specific, and 'off the shelf' re-use may not often be possible. However, domain-specific languages ease the adaptation of models to new situations. SELES includes a hierarchical library of sample landscape events for different process types to speed model prototyping at the start of a project.
- *Communicability*: The declarative nature of our modelling language, our graphical user interface and our related visualization tools (Carpendale et al., 1998) improve communication of model dynamics. However, diagrammatic model construction has the potential to extend the range of people involved in a collaborative project and the ability to communicate complex model dynamics (Maxwell and Costanza, 1997a). We are currently working on a graphical editor for viewing and modifying landscape events.

The need for domain-specific languages for modelling landscape dynamics extends beyond a single framework such as SELES, which represents just one among many possible approaches. The extension of other domain-specific languages, such as SME, SPAMOD, Petri-nets and PCRaster (Gao, 1996; Wesseling et al., 1996; Maxwell and Costanza, 1997a; Groenwold and Sonnenschein, 1998), can provide a range of solutions to the many landscape-scale problems and complement parallel proposals for individual-based modelling (Lorek and Sonnenschein, 1999). For modellers to be able to select the tools best suited to their problem, however, a formal comparison of the strengths and weaknesses of available domain-specific systems for tackling models of different types would be very useful.

Spatial models of landscape dynamics are likely to become increasingly important to help address many of the large-scale land and resource management problems facing society. There is a need for tools that can help to transform conceptual models into implementations that can be simulated on computers. Domain-specific languages support a wide array of modelling tasks, while providing a framework that allows modellers to focus on conceptual model development by separating model specification from implementation. SELES provides model-level support for a class of dynamic spatial landscape models without making specific assumptions about the particular models to be implemented. Our high-level modelling language allows the dynamics of spatial processes to be clearly, formally and concisely stated without the details of how this behaviour is to be executed at the procedural level. Our structured environment supports model development at a level closer to the conceptual model, allowing ecologists and planners to focus on the description of ecosystem and management processes of interest.

Acknowledgements

This research was partially funded by a grant from Forest Renewal British Columbia. We are thankful for the use of facilities provided by Simon Fraser University, and for assistance by

other members of this project, in particular Dr. Sheelagh Carpendale, David Cowperthwaite and Dr. F. David Fracchia in the School of Computing Science, and Dr. Ken Lertzman in the School of Resource and Environmental Management. We also thank Don Morgan, of the B.C. Forest Service, and Dave Daust for collaborating on the Iskut-Stikine landscape model. We are grateful to Glenn Sutherland, Brigitte Dorner and two anonymous reviewers for comments that helped to greatly improve this paper.

References

Acevedo, M.F., Urban, D.L., Shugart, H.H., 1996. Models of forest dynamics based on roles of tree species. Ecol. Model. 87, 267–284.

Allen, T.F.H., Hoekstra, T.W., 1992. Towards a Unified Ecology. Columbia University Press, New York 384 pp.

Baker, W.L., 1989. A review of models of landscape change. Land. Ecol. 2, 111–133.

Baker, W.L., 1992. Effects of settlement and fire suppression on landscape structure. Ecology 73, 1879–1887.

Baltzer, H., Braun, P., Koehler, W., 1998. Cellular automata models for vegetation dynamics. Ecol. Model. 107, 113–125.

Barber, J., 1992. The need for application-specific software tools. Computer Design 31, 116.

Boyce, M., 1996. Review of RAMAS/GIS. Quart. Rev. Biol. 71, 167–168.

Boychuk, D., Perera, A.H., Ter-Mikaelian, M.T., Martell, D.L., Li, C., 1997. Modelling the effect of spatial scale and correlated fire disturbances on forest age distribution. Ecol. Model. 95, 145–164.

CACI, 1987. SIMSCRIPT II.5 programming language. CACI Products (www.caciasl.com/simscript.html), La Jolla, CA.

Carpendale, S., Cowperthwaite, D., Tigges, M., Fall, A., Fracchia, F.D., 1998. The Tardis: a visual exploration environment for landscape dynamics. In: Visual Data Exploration and Analysis VI Conference. San Francisco, CA.

Clark, J.D., 1992. Modeling and simulating complex spatial dynamic systems: a framework for application in environmental analysis. Simul. Dig. 21, 9–19.

Costanza, R., Duplisea, D., Kautsky, U., 1998. Ecological modelling and economic systems with STELLA. Ecol. Model. 110, 1–4.

Derry, J.F., 1998. Modelling ecological interaction despite object-oriented modularity. Ecol. Model. 107, 145–158.

Desmet, P., Govers, G., 1995. GIS-based simulation of erosion and deposition patterns in an agricultural landscape: A comparison of model results with soil map information. CATENA 25, 389–401.

de Vasconcelos, M., Zeigler, B., 1993. Discrete-event simulation of forest landscape response to fire disturbance. Ecol. Model. 65, 177–198.

Dunning, J., Stewart, D., Danielson, B., Noon, B., Root, T., Lamberson, R., Stevens, E., 1995. Spatially explicit population models: current forms and future uses. Ecol. Appl. 5, 3–11.

Fall, A., 1998a. The foundations of taxonomic encoding. Comput. Intell. 14, 598–642.

Fall, A., Fall. J., 1999a. Beauty and the beast: separating specification from implementation for models of landscape dynamics. Simon Fraser University Technical Report SFU CS TR 99-06.

Fall, A., Fall. J., 1999b. SELES user and modeller documentation. Available on the web at http://www.cs.sfu.ca/research/SEED.

Fall, J., 1998b. Reconstructing the historical frequency of fire: A simulation approach to developing and testing methods. MRM thesis, Simon Fraser University, Vancouver, B.C.

Forrester, J.W., 1961. Industrial Dynamics. MIT Press, Cambridge, MA.

Frelich, L.E., Lorimer, C.G., 1991. A simulation of landscape-level stand dynamics in the northern hardwood region. J. Ecol. 79, 223–234.

Gao, Q., 1996. Dynamic modeling of ecosystems with spatial heterogeneity: a structured approach implemented in Windows environment. Ecol. Model. 85, 241–252.

Gardner, R.H., O'Neill, R.V., Turner, M.G., Dale, V.H., 1989. Quantifying scale-dependent effects of animal movement with simple percolation models. Land. Ecol. 3, 217–227.

Grant, W.E., French, N.R., 1990. Response of alpine tundra to a changing climate: a hierarchical simulation model. Ecol. Model. 49, 205–227.

Groenwold, A., Sonnenschein, M., 1998. Event-based modelling of ecological systems with asynchronous cellular automata. Ecol. Model. 108, 37–52.

Itami, R., 1994. Simulating spatial dynamics: cellular automata theory. Land. Urban Plan. 30, 27–47.

Klenner, W., Kurz, W.A., Webb, T.M., 1997. Projecting the spatial and temporal distribution of forest ecosystem characteristics. In: Proc. GIS 97. GIS World, Fort Collins, CO, pp. 418–421 www.essa.com/forestry/telsa.

Krauchi, N., 1995. Application of the model FORSUM to the Solling spruce site. Ecol. Model. 83, 219–228.

Lertzman, K., Fall, J., Dorner, B., 1998. Three kinds of heterogeneity in fire regimes: at the crossroads of fire history and landscape ecology. Northwest Sci. 72, 4–23.

Lorek, H., Sonnenschein, M., 1998. Object-oriented support for modelling and simulation of individual-oriented ecological models. Ecol. Model. 108, 77–96.

Lorek, H., Sonnenschein, M., 1999. Modelling and simulation software to support individual-based ecological modelling. Ecol. Model. 225, 199–216.

Maxwell, T., Costanza, R., 1997a. A language for modular spatio-temporal simulation. Ecol. Model. 103, 105–113.

Maxwell, T., Costanza, R., 1997b. An open geographic modeling environment. Simul. J. 68, 175–185.

Merkuryeva, G., Merkuryev, Y., 1994. Knowledge based simulation systems — a review. Simulation 62, 74–89.

Minar, N., Burkhart, R., Langton, C., Askenazi, M., 1996. The Swarm simulation system: a toolkit for building multi-agent simulations. www.santafe.ed/projects/swarm.

Mladenoff, D.J., Host, G.E., Boeder, J., Crow, T.R., 1996. LANDIS: a spatial model of forest landscape disturbance, succession and management. In: Goodchild, M., Steyaert, L.T., Parks, B.O. (Eds.), GIS and Environmental Monitoring. GIS World Books, Fort Collins, CO, pp. 175–179.

Moloney, K.A., Levin, S.A., 1996. The effects of disturbance architecture on landscape-level population dynamics. Ecology 77, 375–394.

Olde, V.H., Wassen, M.J., 1997. A comparison of six models predicting vegetation response to hydrological habitat change. Ecol. Model. 101, 347–361.

O'Neill, R.V., DeAngelis, D.L., Waide, J.B., Allen, T.F.H., 1986. A Hierarchical Concept of Ecosystems. Princeton University Press, Princeton, NJ.

Province of British Columbia, 1995. Biodiversity Guide Book. Forest Practices Code of British Columbia. B.C. Ministries of Forests and Environment, Lands and Parts, Victoria, B.C.

Sklar, F.H., Costanza, R., 1991. The development of dynamic spatial models for landscape ecology: A review and prognosis. In: Turner, M.G., Gardner, R.H. (Eds.), Quantitative Methods in Landscape Ecology. Springer, New York, pp. 239–288.

Turner, M.G., 1988. A spatial simulation model of land use changes in a piedmont county Georgia. Appl. Math. Comput. 27, 39–51.

Turner, M.G., 1989. Landscape ecology: the effect of pattern on process. Annu. Rev. Ecol. Syst. 20, 171–197.

Turner, M.G., Gardner, R.H., Dale, V.H., O'Neill, R.V., 1989. Predicting the spread of disturbance across heterogeneous landscapes. Oikos 55, 121–129.

Wallin, D.O., Swanson, F.J., Marks, B., 1994. Landscape pattern response to changes in pattern generation rules: land-use legacies in forestry. Ecol. Appl. 4, 569–580.

Wesseling, C.G., Karssenberg, D., Van Deursen, W.P.A., Burrough, P.A., 1996. Integrating dynamic environmental models in GIS: the development of a dynamic modelling language. Trans. GIS 1, 40–48.