FOOD AND ECOLOGICAL SYSTEMS MODELLING JOURNAL

Food and Ecological Systems Modelling Journal 5: e121215 (2024)
DOI: https://doi.org/10.3897/fmj.5.121215

Methods

ALMaSS Landscape and Farming Simulation: software classes and methods

Christopher John Topping[1], Xiaodong Duan[1]

1 *Social-Ecological Systems Simulation Centre, Department of Ecoscience, Aarhus University, Aarhus, Denmark*
Corresponding author: Christopher John Topping (cjt@ecos.au.dk)

Abstract

This article describes the landscape model for the Animal Landscape and Man Simulation System (ALMaSS). The article itself provides an overall description of the software classes and functions, with examples of use. In addition, there are links for code access, executable directory and code documentation for the model. These provide access to model details.

The descriptions cover the landscape class, its internal data, and the class hierarchy primarily associated with the landscape simulation. They also cover the timing and scheduling of actions and provide an overview of the landscape class interface. The Farm and Farm Manager classes responsible for simulating farming activities are presented together with descriptions of crop classes for crop management and weather and calendar classes.

Key words: ALMaSS, OOP, Vegetation Growth Modelling

Introduction

The Animal Landscape and Man Simulation System (ALMaSS) (Topping et al. 2003) was developed in 1996 and first used in Danish projects in 2000. ALMaSS was a response to a request from the Danish Ministry for Energy and Environment to be able to answer unspecified questions related to the impact of human management on wildlife (Topping 2022). The underlying design of the landscape and associated classes has been rather constant over time and has been referenced in part in several publications over the years (e.g. Topping et al. 2003; Topping et al. 2018). However, a full description covering the landscape simulation components has not been provided, and recent updates to the system provide an opportunity to rectify this. The landscape and associated classes are large, and implementation in code is not trivial; as such, the descriptions provided in this overview are quite general. The aim is to communicate the approach rather than the details, which can be seen in the open-source code available from GitLab and linked to Suppl. material 1.

In ALMaSS, the Landscape class essentially provides the environment into which other models can be run. It can, however, be used as a model in its own right, for example, to generate pesticide use maps. As a consequence of its role, the landscape must be able to supply all information needed by any other model in space and time. Other models thus define the role of the landscape

OPEN ACCESS

Academic editor: Leonardo Candela
Received: 19 February 2024
Accepted: 14 June 2024
Published: 27 June 2024

Citation: Topping CJ, Duan X (2024) ALMaSS Landscape and Farming Simulation: software classes and methods. Food and Ecological Systems Modelling Journal 5: e121215. https://doi.org/https://doi.org/10.3897/fmj.5.121215

Copyright: ©
Christopher John Topping & Xiaodong Duan
This is an open access article distributed under terms of the Creative Commons Attribution License (Attribution 4.0 International – CC BY 4.0).

class, but equally, this means that the full range of detailed data and mechanisms is beyond the scope of a single paper.

There are two parts of the landscape model where two approaches can be taken. These areas are the implementation of pesticide use and its fate and the implementation of farm management. In these cases, the functionality described here is the basic functionality normally used when neither the pesticides nor the farm management are the objects of study, i.e. typical ALMaSS use. Specific cases of detailed pesticide implementation and use (e.g. Topping and Luttik 2017; Topping et al. 2018; Ziółkowska et al. 2021) and detailed farm management behaviour (e.g. Malawska et al. 2014; Malawska and Topping 2016, 2018) can be consulted for details of these approaches. The pesticide implementation in ALMaSS is described by (Poulsen et al. 2023).

In this paper, we describe the steps in linking external data to the internal representation in ALMaSS, the class structure implemented, and the processes covered by the core components of the ALMaSS landscape model.

Methods

Overall design considerations

ALMaSS needs to simulate landscapes of areas up to 50 × 50 km, with a typical size of 10 × 10 km, and at 1 m^2 resolution. The ALMaSS system and the landscape simulation part should support multiple models running simultaneously, and the modelled entities can be in the order of 50 million concurrent agents. Hence, computational efficiency in both memory and CPU speed is critical. However, at the same time, the system must be flexible to support the needs of models ranging from simple insects to human decision-making. To achieve efficiency and flexibility, ALMaSS uses a strong object-oriented paradigm (OOP) in developing the code and relies on C++ development using relatively low-level programming code for efficient coding of algorithms and data structures. The OOP paradigm also confers an important additional benefit: facilitating the management of a huge code structure. OOP in C++ relies on the definition of classes to describe the behaviour of entities in the code. From these classes, descendent classes can be created, inheriting the functionality of the parents. This compartmentalises the code and allows modification of the classes without interfering with the functioning of the wider program. Classes interact with other entities through defined interfaces; thus, this is the only part of the class that needs to be cleanly integrated with the rest of the program. This approach also means new classes can be added without needing a major rewrite, either by extension of previous classes or modification of the interfaces of related components to allow their integration. In the following description, we refer to this class hierarchy to describe the major parts of the landscape simulation.

The Landscape class

The Landscape Class is a fundamental component of ALMaSS, providing the interface for all environmental information required to run agent-based and subpopulation models. It effectively separates external data inputs from the implementation and internal Landscape interface.

Landscape initialisation

The Landscape Class is initialised at the start of any ALMaSS simulation. This initialisation process sets up the primary data structures and classes used by the simulation, playing a crucial role in the ALMaSS simulation system. The key aspects of this initialisation process include:

1. Map Representation - Creating an internal representation of the map and associated classes to manage spatial data and relationships efficiently.
2. FarmManager Setup - Establishing the FarmManager class and initialising all farm-related information, including farm locations, characteristics, and management practices.
3. Behavioural Functions - Implementing miscellaneous functions associated with agent behaviour, such as movement patterns, resource utilisation, and interactions with other agents and the environment.
4. Data Modification and Output Generation - Providing functionalities to modify input data as needed during the simulation and generate output data for subsequent analysis and visualisation.

The Landscape Class is the backbone of the ALMaSS simulation environment, ensuring that all agents and subpopulations can access the necessary environmental information and interact realistically within the simulated landscape.

Creation of internal map data structure

The method used to represent the data as a map inside ALMaSS uses the flyweight design pattern (Gamma et al. 1994). This algorithm saves a great deal of space in ALMaSS by representing the landscape as a set of internally homogenous polygons (e.g., a 1 Ha field). Thus, the flyweight implementation avoids complex polygon calculations, e.g., for the movement of animals, by mapping the polygons on a raster. Each element of the raster map is then a reference to the polygon where the complex descriptive data is stored. Thus, a 1Ha field, it would use 100 × 100 cells in the raster, each with a value pointing to that field's data; this is one reference per m^2. The field data would contain multiple elements such as the vegetation type, height, cover, biomass, farm manager, spilt grain etc.. Storing all this data at each raster element would be highly memory-consumptive since all data would be duplicated at every m^2. Efficiency is further improved because the field polygon data is stored only in one place; therefore, updating this is efficient in CPU time. The class structure used to support this is presented in detail below. In this structure, each polygon is an instance of a class derived from the LE Class (Landscape Element). LE is part of a class hierarchy, so a polygon can exhibit specific behaviour depending upon its descendent type (e.g., the Field class inherits vegetative characteristics from the Veg_Element class, which inherits all behaviour common to all landscape elements from the LE class, then it adds specific field polygon behaviour).

The map structure is created based on the information passed from the PolyRef and .lsb files resulting from the external data handling procedure.

The .lsb file is a landscape binary format file that lists the size and location of the landscape map and holds the reference number to the polygon that covers the majority of the cell for every 1 m^2 cell. This number is linked to the PolyRef file. The .lsb file, therefore, provides the topographic map of the landscape and is read into ALMaSS to create the flyweight data structure described above.

The format of the PolyRef file is to list one row for each polygon in the landscape and for the area covered by these to have 100% coverage for the rectangle describing the landscape. An example of the top of a PolyRef file is shown in Table 1. Example of a polygon reference (PolyRef) file input from a Danish landscape denoted by DK in the first row which is followed by the location (Latitude and Longitude) of the NE corner of the landscape that this file refers to. The second row is the number of polygons defined in the file. Polytype is a standard ALMaSS code for this polygon type. PolyRefNum is a unique identifier for the polygon in this map. Area is the area in m^2. FarmRef is a reference to the farm that owns the polygon, if any (or -1 if there is not farm associated with it). Unsprayed Margin Ref indicates the PolyRef for the associated unsprayed margin, if any (or -1). Soiltype is the soil type under the polygon as the most common type, or -1 if not known. Centroid X and Centroid Y are the coordinates of the centroid of the polygon. Openness, Elevation, Slope and Aspect are not used in this example but can be included if needed.1. The PolyType denotes the TypeOfLandscaleElement (TOLE) class this polygon belongs to, and the PolyRefNum

Table 1. Example of a polygon reference (PolyRef) file input from a Danish landscape denoted by DK in the first row which is followed by the location (Latitude and Longitude) of the NE corner of the landscape that this file refers to. The second row is the number of polygons defined in the file. Polytype is a standard ALMaSS code for this polygon type. PolyRefNum is a unique identifier for the polygon in this map. Area is the area in m^2. FarmRef is a reference to the farm that owns the polygon, if any (or -1 if there is not farm associated with it). Unsprayed Margin Ref indicates the PolyRef for the associated unsprayed margin, if any (or -1). Soiltype is the soil type under the polygon as the most common type, or -1 if not known. Centroid X and Centroid Y are the coordinates of the centroid of the polygon. Openness, Elevation, Slope and Aspect are not used in this example but can be included if needed.

DK Latitude 57.3801 Longitude 9.83159											
41058											
PolyType	PolyRefNum	Area	Farm Ref	UnSprayed Margin Ref	SoilType	Openness	CentroidX	CentroidY	Elevation	Slope	Aspect
123	0	165	-1	-1	-1	0	260	17	0	0	0
130	1	311	-1	-1	3	0	366	44	0	0	0
130	2	195	-1	-1	-1	0	421	23	0	0	0
13	3	1805	-1	-1	3	0	442	647	0	0	0
13	4	25	-1	-1	-1	0	430	9	0	0	0
210	5	289	-1	-1	-1	0	438	14	0	0	0
210	6	313	-1	-1	-1	0	536	31	0	0	0
98	7	224	-1	-1	-1	0	656	11	0	0	0
207	8	55	-1	-1	-1	0	654	13	0	0	0
210	9	8606	-1	-1	3	0	731	23	0	0	0
110	10	1172	-1	-1	3	0	874	13	0	0	0

is a unique (for this map) identifier for this polygon. Normally, after an initial ALMaSS load and run, these will be renumbered automatically to be 0 to one less than the number of polygons. All other columns are attributes of the particular polygon. The FarmRef is of key importance. This reference indicates either -1 if this polygon is not part of a farm or a reference number of a farm to which the polygon belongs and is managed. UnSprayedMarginRef indicates whether the polygon (if it is a field) has an unsprayed field margin, and SoilType denotes the soil type as an integer reference (Table 2. ALMaSS soil types as referenced inside ALMaSS code with their FAO classification, agricultural usage and class for management of crop management.2). The other attributes denote its elevation, slope and aspect, and openness. Openness is a measure of distance to the nearest tall or large artificial object from the polygon's centre point (CentroidX & CentroidY). These values are optional, but ALMaSS will always create the centre point on the first run for a new landscape. The PolyRefNum becomes the values (starting from 0) stored in the raster to support the flyweight algorithm. The .lsb file is essentially a raster map of PolyRefNum at 1-m resolution in binary format. Thus, when loading the files a representation of the landscape map is loaded to memory and internally linked, via the PolyRefNum, to an object representing each polygon. This object will be descended from the Landscape Element (LE) class (see below), the type dependent upon the TOLE value it contains.

Table 2. ALMaSS soil types as referenced inside ALMaSS code with their FAO classification, agricultural usage and class for management of crop management.

Agricultural Usage	ALMaSS Code SoilType	FAO Texture Code	Description	Crop management types
None	0	-	Water	-
None	1	S	Sand (unspecified)	Light (sandy type)
Poor	2	LS	Loamy swand	Light (sandy type)
Poor	3	SL	Sandy loam	Light (sandy type)
Average	4	SCL	Sandy clay loam	Light (sandy type)
Average	5	SiL	Silt loam	Average (silty loam type)
Good	6	SiCL	Silty clay loam	Average (silty loam type)
Good	7	CL	Clay loam	Heavy (clay & loam type)
Good	8	L	Loam	Heavy (clay & loam type)
Good	9	Si	Silt	Average (silty loam type)
Good	10	SC	Sandy clay	Heavy (clay & loam type)
Good	11	SiC	Silty clay	Heavy (clay & loam type)
Good	12	C	Clay	Heavy (clay & loam type)
Good	13	HC	Heavy Clay	Heavy (clay & loam type)
Good	14	O	Organic & peat	Organic & peat

Landscape class Scheduling

The basic timestep for landscape operations is one day. This means that several operations must be carried out at the start of each simulation day. These collectively update the landscape simulation so the other ALMaSS models can access the data specific to that day on completion. These operations fall into four main types, 1) updating the calendar, 2) updating the weather, 3) updating

the farming, and 4) updating the polygon internal behaviour (through updating all LE objects). Each of these four areas are detailed below. Date and weather are controlled by the Calendar and Weather classes, respectively, and are described in their respective sections. Farming is a complex subject area of the landscape simulation and is detailed below (See Farm, FarmManager and Crop classes), however, the landscape is responsible for calling the FarmManager class daily and asking it to update all the farms. The Landscape controls the link between updating the polygons and the Landscape class by calling the 'Tick' function for each polygon. The Tick function can be implemented differently for different base TOLE types within the LE class hierarchy but is executed correctly by OOP polymorphism.

The Landscape class interface

The Landscape Class has an extensive interface function set, which provides the rest of the simulation with access to landscape data or data from LE instances or Farms. Using this interface rather than direct access to the data simplifies the maintenance of the code by allowing the implementation within the Landscape and associated classes to be altered without affecting the rest of the simulation models.

LE class hierarchy

The LE class (Landscape Element) is the base class for describing polygons in the landscape. Descendent classes implement specific behaviour for different TOLE types. This hierarchy (Fig. 1: The landscape element class hierarchy based on the LE base class. All types of landscape element are classified as being in one of these classes. Those with specific behaviour are placed in the lowest tier whilst those represented by generic types only differing in attribute values are placed in the NonVegElement and VegElement class.1) uses the polymorphism concept of OOP, with most of the functionality of the descendent classes being implemented in base classes but specific details implemented in descendants. However, relying on this class structure and the TOLE types limits efficiency as the total number of TOLE types is quite high (currently 117, with multiple TOLE types represented by individual LE descendent classes). Consequently, many class definitions, bloating code, and relatively long conditional statements would be required to use the TOLE type or class type as a viable way to work with habitats in the landscape.

The problem of multiple types is avoided by adding classificatory attributes and the relatively short class hierarchy in Fig. 1. The main classificatory attribute is the TOLE type, thus a CorkOakForest would be a LE->VegElement->ForestElement class, with the attribute of type CorkOakForest. Creating and assigning further attributes is flexible, but standard attributes denote wooded vegetation, forest, grassland, urban, and tall vegetation. This classification based on TOLE attributes is continued with a further detailing of vegetation types. TOVs (Type of Vegetation) are used to identify the vegetation type present on a polygon. For some types this is fixed e.g., a grazed permanent grassland will always be derivative of TOV PermanentGrassGrazed, but for others it is the management that decides the vegetation type (e.g., for arable fields with crop rotation).

Similar to TOVs, classification attributes are used for TOVs (of which there are currently 321) to make identification of particular types easy, e.g., grassland, cereal, maize. These attributes allow, e.g., a vole model, to interrogate a polygon to determine if it is a potentially suitable habitat by asking if it's grassland, rather than having to ask whether the polygon is each type of grassland included in the TOLE and TOV types.

Flexibility is further increased through 'user-defined' attributes that individual models can use to create specific combinations of TOLE and TOVs. For efficiency, these attributes are defined at run-time and written into the attributes of polygons on creation of the Landscape instance. The LE and descendent classes have a wide range of functionality. For example, the road classes can represent the traffic loads at any given point in time.

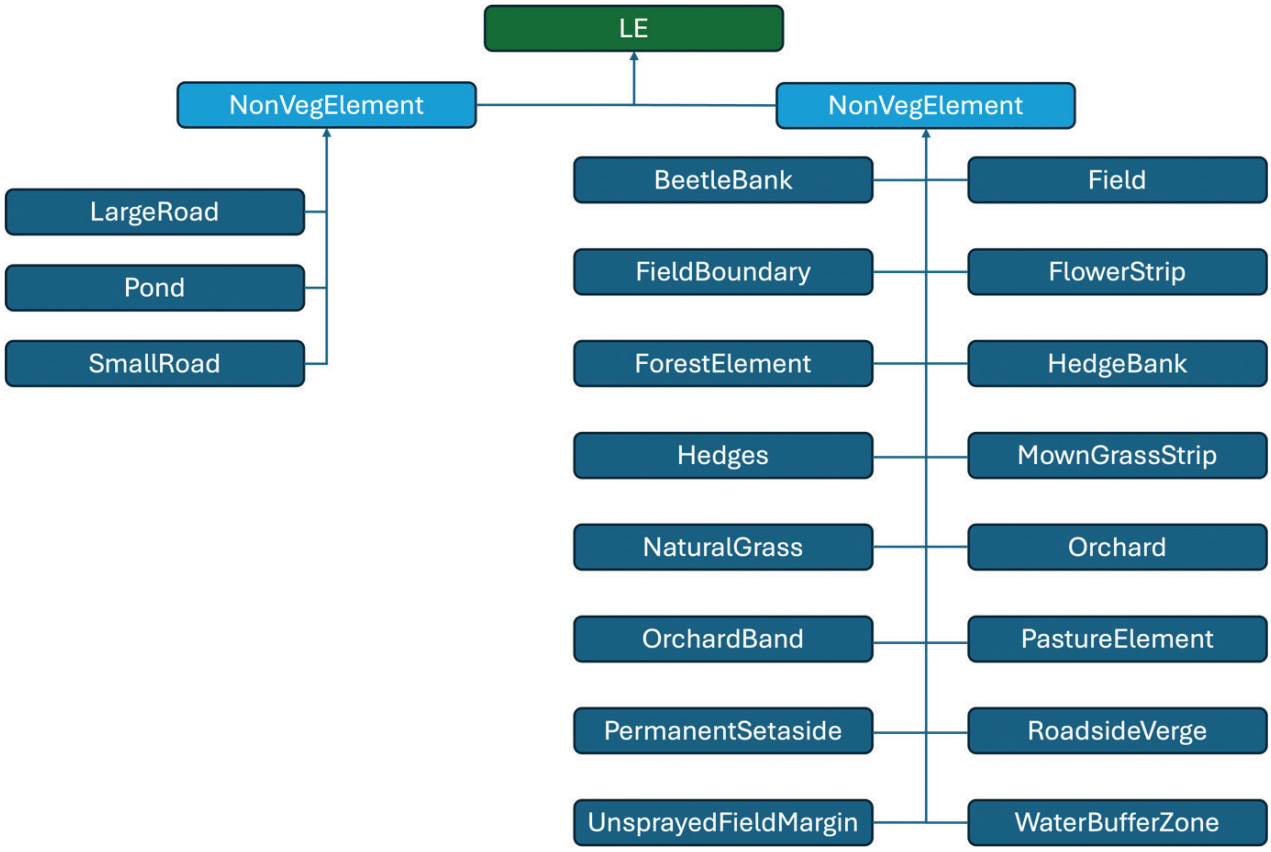

Figure 1. The landscape element class hierarchy based on the LE base class. All types of landscape element are classified as being in one of these classes. Those with specific behaviour are placed in the lowest tier whilst those represented by generic types only differing in attribute values are placed in the NonVegElement and VegElement class.

A major part of the LE functionality is modelling vegetation growth (see below). Other LE functionality is targeted at supporting specific other models. There is a sub-model for general insect biomass, others for decay rates of spilled grain and good forage resources, digestibility of vegetation, and a new set of functionality under development for supporting pollinator resource modelling. The design here relies on the sensible inclusion of only those functionalities actually in use for any given simulation, following the concept of avoiding false inclusions but also false exclusions (Topping et al. 2015). This means that if a field vole simulation is running there is no need to calculate the insect

biomass, but that calculation of vegetation digestibility is needed instead. This behaviour is a dynamic choice taken by the simulation based on the number and types of models extant after start-up.

As mentioned above, some LE descendent classes have special behaviours. These are largely related to management activities, such as mowing roadside verges or cutting grass. Of these, the Field class has a specific role in providing functionality related to rotational crops. This functionality includes recording the current, past and planned next crop and some special behaviour simulating the creation of tramlines when the tractor drives into the field and their subsequent closure as vegetation grows.

Vegetation growth modelling

One key part of implementing the external data in the simulation is the simulation of vegetation growth. Any class with VegElement as the base class will simulate vegetation development on a daily basis. The underlying method is described by (Topping and Olesen 2005) and relates the development of vegetation in terms of height and leaf area index (LAI) to the day-degrees experienced since sowing (or the beginning of the year). Both green LAI and total LAI are calculated. Multiple vegetation curves are defined for ALMaSS, each represented by a set of consecutive inflection points and gradients of change per day degree between each of these points.

Further flexibility is obtained by creation of different growth phases, describing different periods of the plant growth or responses to external influences. Five phases are currently implemented, and can be used when defining a growth curve as needed. These are 1) From January 1st; 2) From Sowing; 3) From March 1st; 4) After Harvest/cut; 5) After Second Harvest/cut. Complex patterns can be simulated by combining the growth curve phases differently. For example, an autumn sown wheat crop might have: From sowing in September, followed by January to March, followed by From March, followed by After Harvest. Each TOV type has to have an associated plant growth curve, but this curve may be used by multiple TOVs (e.g., most variants of barley have the same growth curve). During the simulation, major management events such as harvest or cutting events will trigger an immediate jump to a different growth phase. In this way the flexible timing of the crop management can be taken into account.

LAI is converted to cover through application of Beer's Law to calculate vegetation cover. The equation using Beer's law to calculate the cover (C) $C = 1-\exp(-k*\mathrm{LAI})$ where k is the light extinction coefficient and is assumed for cover to be 0.6, and LAI is the total LAI. This value is towards the high end of the range for winter wheat (Soleymani 2016) and was chosen as a reasonable estimate for most vegetation. Change in biomass is implemented as described by (Topping and Olesen 2005) using the green LAI, the amount of radiation and a k of 0.4. Biomass, height, and LAI values are also further used to determine species-specific properties of vegetation, e.g., the vegetation digestibility experienced by a hare (Topping et al. 2010). Deviations in vegetation biomass due to other interventions e.g., grazing, result in a slow regrowth of the vegetation towards the point it would have reached with no intervention.

Farm, FarmManager and Crop classes

Agricultural simulation is a large part of the Landscape functionality. This is achieved by defining a hierarchy of interacting objects starting with a FarmManager class which controls all the lists of farms and their scheduling, the Farm class which represents instances of individual farms, and the Crop class which defines the management of each crop grown on farm fields.

The Farm Manager class

This class is an administrative class controlling the list of farms and is involved in scheduling farming actions. This is called once per day from the Landscape as part of the daily updating of the landscape simulation. The FarmManager is created with the Landscape class from the FarmRefs.txt file generated during the external data handling process. This file contains a list of all farms together with a reference for their farm type, e.g., conventional arable farm, and is used by the FarmManager to create and populate the list of farm objects in the simulation. After the creation of the farms the FarmManager creates a centre point for each farm (centroid) which can be used by other objects in the simulation. The FarmManager class itself has no further functionality but acts as an interface to farm information, providing lists and data as necessary to support simulation elsewhere in ALMaSS, often using the farm reference number from the FarmRefs.txt as an index to the farm of interest.

The Farm class

The Farm Class is the organising class for each individual farm and its data. The Farm object is created, and its management is initialised by the FarmManager on start-up. Each farm consists primarily of a list of fields for which it is the owner/manager. Some of these fields may be designated as permanent crops (e.g. pasture or trees such as orchards), and others will be fields in a crop rotation. Depending on the level of information present in the external data sources, permanent crops may be identified individually and exist as separate TOLEs in the map or may be represented by a proportion of the farm area. In the latter case, the allocation of permanent crops is made to the farm fields to best match the proportion of area under those crops reported to the CAP subsidy scheme used to create the underlying data. On construction, the farm object will be linked to all its fields via a list of pointers. The farm will also be allocated a Farmer object, but this class is not needed to run the standard ALMaSS landscape simulation.

After the management's initialisation, the Farm object will allocate a crop to each field. Permanent crops have a fixed vegetation type, but rotational crops must be chosen. In the rotational case, each field will be randomly allocated one crop from the crop rotation for the farm (which is a function of the farm type). This rotation is typically 100 crops long and represents the area covered by each crop type in one-percentage blocks. The crop rotation needs to be specified by the user and associated with the farm type. This is not an automatic process but uses external data as the basis for developing a '.rot' file, i.e., the list of crops in the rotation. Before a simulation can be run a .rot file must exist and be associated with each farm type present in the landscape.

Each crop type (at TOV) has a specific management that is defined by a crop management code file. This management causes the execution of farm activities such as ploughing. These farm activities (called FarmToDo in ALMaSS) are a property of the Farm class. Each FarmToDo has the following basic behaviour when called:

- Check whether conditions are suitable for the operation, if not exit
- Apply any changes to vegetation structure e.g., remove it for ploughing, cut to 10 cm for harvest.
- Apply any mortalities to the general insect population model that may be needed
- Record that this type of operation was carried out on the field polygon in a history of events
- Record that this was the last FarmToDo carried out
- Repeat these measures for any associated unsprayed field margins (counted as separate polygons).

Storing the last FarmToDo as an easily accessible 'last event' and a traced history provides maximum flexibility for other models to access management history.

Crop classes

Crop classes hold the information necessary to manage a farmed crop. This may be an arable crop in rotation or a permanent crop, including grazed grassland. The basis for the development of a crop class is a table of events and a resulting flow diagram which describes farming operations with conditions for implementation (e.g., related to weather or crop growth) and periods during which they are implemented. The diagram follows 100% of farmers starting from any post-harvest operations from the previous crop until the end of harvest or any post-harvest operations for this crop. At each decision branch, a certain proportion of farmers will be allocated to each possible path. If this allocation is based on probability, the overall consistency over time is maintained, meaning that if 100% of farmers start, 100% will also reach the end of the plan.

There is only one instance of each crop class. This is done to encapsulate the field polygon information in the LE class instance (i.e., the field where the crop is grown). In this way, duplicate data is avoided, and it is clear that any cropping information needed is accessed via the LE class interface. All functions of the crop class are contained in the single instance of that class, which means that all necessary data to run each crop on each field must be passed to this crop object each time it is called. This data is passed using an event queue containing pointers to the farm owner object and the field through which the data can be accessed. When a crop is started on a field, an initiation event is sent to the queue; this is a timed event that may be for the next time step or months or more. When this event is triggered, the crop management code starts. The farm management to be carried out then generates new events placed on the stack in the same way; they can also fork multiple events from a single point. The management plan is thus very flexible

and runs as a multi-threaded program. Only one thread (designated the main thread) reaches the end of the plan's execution, returning a 'Done' signal on completion, which then triggers choosing the next crop in the rotation for that field and sending a new initiation event to the queue. The other threads may end or re-join the main thread (Fig. 2). Very complex management plans can be constructed using this method. Each farm event may have a probability of being applied and multiple dependencies on field information, such as soil or the history of events for that field and season. This control structure results in the crop management becoming dynamic depending on the farming context, which changes during the simulation run.

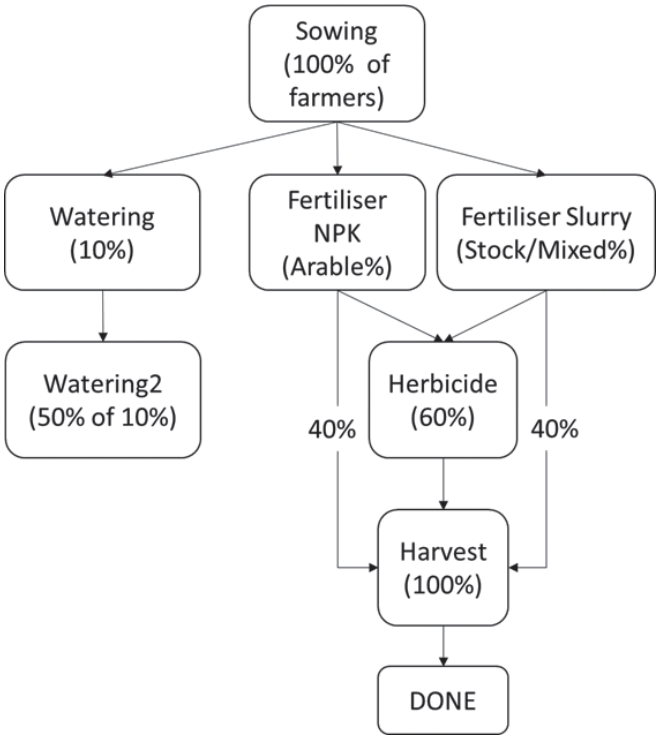

Figure 2. A simple example crop management scheme with parallel threads of execution, one of which is dependent on farm type (arable or stock farms). The watering thread terminates and all other threads lead to harvest.

The range of conditions related to branching in the crop is large. These may link to the farm type as in the example in Fig. 2: A simple example crop management scheme with parallel threads of execution, one of which is dependent on farm type (arable or stock farms). The watering thread terminates and all other threads lead to harvest.2, but equally may be related to vegetation characteristics, weather conditions, soil conditions, simple probability or a combination of these. A common controlling structure is to use conditions based on whether previous operations have been done e.g., if pesticide was just sprayed do not water, or if 'herbicide treatment 1' was carried out, consider 'herbicide treatment 2'. The overall result is a very flexible and descriptive program structure.

There is an interplay between the vegetation growth curves and management plans. Some events, such as harvest, will trigger a change in the vegetation growth phase, thus altering crop growth.

Weather and Calendar classes

ALMaSS does not simulate weather, but instead, it's an input. Weather data consists of hourly mean temperature, precipitation, wind speed, wind direction, soil temperature, snow depth, humidity, and radiation. Weather is based on a historical record for an area and is part of the external data procedure needed for ALMaSS. Because it is a historical record, it is usually looped for long simulations. Consequently, models in the system can request weather information for periods during the simulation, even for the future. For example, a crop code may trigger a farm management event requiring five days without rain before execution.

The Weather Class interface provides access to all weather data as immediate weather data (at the point in time the function is called), at a date, particular hour, day, or over a specific period. Thus, the weather class can be viewed as a non-dynamic data supplier.

The Calendar class performs timekeeping for ALMaSS. Depending on the time step used (typically 1 day to 10 minutes for animals), the Calendar class will provide the current time, as well as general functionality related to the number of days per month and indexing for time-dependent function calls (e.g., supplying rain over a period of time). The calendar class is a critical part of the crop management functionality since it supplies Julian days and the simulation date, both used in the Crop Class codes.

The Calendar class has a detailed set of functions for calculating the sunrise and sunset time. This calculation is done automatically from the latitude and longitude supplied during the landscape generation process. This position is used to calculate the time at which the sun drops below or comes above the horizon based on an angle (90° for official and 108° for astronomical).

Outcomes

ALMaSS landscape simulation attempts to find a balance between the inclusion of necessary detail to support the other system models, flexibility to change or include more, and efficiency at run-time. A further consideration is that ALMaSS was designed for ecologists, not software engineers, and the majority of contributors to the models are not professional programmers. Consequently, code should be written as simply as possible to maximise understanding.

Using the approach described above allows ALMaSS to simulate a landscape in considerable detail. A 10 × 10 km landscape in Denmark typically has upwards of 40,000 polygons and 150–200 farms represented. Simulation runtime for the landscape with no animals is fast, approximately one year per second on an average PC. During this year, all farm operations are carried out daily, and all vegetated elements model their vegetation biomass, LAI, and height daily, as well as digestibility, insect biomass, and other sub-models daily. RAM is a limiting factor, with landscapes of 50 × 50 km requiring a computer with 64GM RAM to run animal simulations. At 50 × 50 km, the raster map alone requires 20GB of memory. However, there is linear scaling between area and run-time for the landscape simulation.

The main aim of the landscape is to serve other simulation components with data requirements. Communication is done via interface functions such as SupplyVegPhase(int a_poly), which returns the current vegetation growth

phase for an LE. Direct polygon calls are the fastest way to access the data with the index to the polygon data array being provided as a_poly. However, direct calls are not always possible, and access has to be via the x,y coordinate system to determine the polygon index. Efficiency here is lower since a_poly must be reached via the raster map using x+(yW), where W is the width of the landscape in m. The need for range checks at run-time for these calls typically results in this being the low-level time-sensitive operation when running an animal model. Despite these problems, ALMaSS can cope with millions of concurrent agents and, for some simulations, up to 50 million.

A key feature of the landscape simulation, as carried out using this modular software process, is the ability to upgrade any component as necessary without compromising the model's integrity. This design allows for the continued testing and development of the code to be possible.

Functionally, the design facilitates the evaluation of complex questions such as spatially differentiated management (e.g., by application through farm type) or complex systems dynamics operating through long-chain interactions, resulting in emergent properties. For example, in Malawska and Topping (2018), the looped chain sequence of policy->farmer->pesticide->insects->birds->farmer->pesticides…. was implemented.

Discussion

The ALMaSS landscape model is a highly detailed landscape simulator that is specifically designed to support ALMaSS model simulations of species and people. As such it does not have a direct analogue in the scientific literature. ALMaSS is designed to represent actual landscape areas, but could be used in conjunction with landscape generators (Langhammer et al. 2019), and indeed has been used to simulate simple geometric and generated maps (Hoye et al. 2012; Topping and Lagisz 2012). In its current form, the model is not designed to predict agricultural yields or work with farm management. For this purpose, there are other specifically designed models or model groups; see Assante et al. (2021) and Terribile et al. (2024) for recent examples of decision-support systems with multiple models supported. Other simulation software systems exist; the SWARM system launched just before ALMaSS was created (Minar et al. 1996). However, SWARM and similar systems are aimed at a lower level of development than ALMaSS, providing the tools and classes to build simulations.

In terms of supporting ALMaSS species and human models, the ALMaSS landscape simulation has been robust, flexible, and reliable for over 20 years. However, many aspects could be improved. Some of these are related to the fact that ALMaSS has grown beyond its initial design parameters, i.e., representing Danish landscapes (Topping 2022), and has become a team activity with many programmers not having formal training or long experience. One key characteristic of this development is code bloat, particularly with the crop management codes. With hundreds of crop management codes, the number of separate crop codes increases each time a new crop is needed for any simulated country. Each code and reference to it must be added to ALMaSS in a number of places, and each must be separately maintained as other changes are made in the system. A solution to this problem would be to move to a domain-specific language (DSL), which exists to describe models for information flow e.g. in

precision farming (Groeneveld et al. 2021). However, this change would require a substantial rewrite as well as defining the DSL itself. As such, it will only be possible in a future version, and this change has no deadline.

A similar issue is connected to the use of the TOLE and TOV types to represent landscape elements and vegetation types. The increasing size of the enumerations for TOLE and TOV types makes using these as control structures less and less efficient. The use of attributes described above goes some way to rectify this, but it is not an easily managed solution. Hence, in future versions, the aim is to create a hierarchical classification system for TOLE and TOV types. Such a classification will facilitate much faster conditional statement execution by limiting the number of queries to the lowest common denominator. However, for this to work, the underlying classification algorithm must map well to the conditional tests that ALMaSS models need. This will refer to function rather than topographical classification, see for example (Cullum et al. 2016). However, the rules for developing this classification are not yet defined.

In the most recent version of the ALMaSS Population_Manager class (Topping and Duan 2024), there is a change to utilise the power of multi-threaded computation using OpenMP (Chandra 2001). This has not been extended to the Landscape and associated classes, but many of the landscape functions are quite suitable for multi-threading since there are few interactions between polygons possible. Multi-threading has not been a priority since the landscape simulation is generally not the rate-limiting step, except for detailed pesticide simulations (Poulsen et al. 2023). However, this feature will be included in future updates.

The ALMaSS landscape has been designed to service the needs of ALMaSS species models, but since it is written in a modular form as a set of linked C++ classes, it could be used as a library to be added to other model systems. However, the easiest way to use it for new models would be to add new models to the ALMaSS framework. Expanding the models available within the system would provide more building blocks for future modellers.

Expansion also has another benefit. The ALMaSS system is designed to answer complex questions about landscape, management, and wildlife. These questions are multi-faceted, often requiring the integration of multiple modules. For example, determining the impact of wildlife management options requires integration of landscape and cropping modules with wildlife management options and multiple species (Topping et al. 2019), and using ALMaSS to evaluate the impact of stressors on wildlife requires integration of similar landscape features but also ecotoxicological modules (e.g. Topping et al. 2018; Ziółkowska et al. 2021). Development in the direction of multi-modular systems is probably a necessary trend in ecology since the questions being asked are more and more on a systems level, requiring more complex models. There are different ways in which these models can be made, but a key requirement is the compatibility of the components to be linked. One way compatibility can be achieved is through multi-modular systems such as the class approach used by C++ and ALMaSS, which effectively creates libraries that can be connected through their interfaces. However, to function properly, the libraries need to be well-documented, version controlled and accessible. This level of documentation and standardisation for large systems requires teams of scientists and programmers and long

time-frames, which is difficult to achieve in academic projects and institutions. Even for smaller models, this documentation is clearly a problem. Janssen et al. (2020) evaluated 7500 publications about individual-based and agent-based models. They found that code availability was maximally 18%, and poor or lacking documentation was a common problem, often leading to a lack of transparency and lower citation rates. However, documentation is not a simple matter. Accuracy and completeness are necessary for software documentation but are not sufficient alone (Treude et al. 2020). To work, documentation must be accessible and carefully designed in appearance and usability. This paper thus lays a foundation for such documentation for the ALMaSS landscape classes, which can be built on using more detailed documentation formats. Here, we have produced the code documentation using doxygen.

ALMaSS Landscape model usage

The ALMaSS landscape simulation model has had significant impacts (over 80 published papers and reports), being successfully utilised in three European H2020 projects: PoshBee, B-GOOD, and EcoStack, and currently supporting the population modelling framework in the Horizon Europe project PollinERA (Topping et al. 2024). It has been employed to evaluate the impact of landscape structure and source-sink dynamics on non-target arthropod pesticide risk assessments in Germany for the German EPA (Ziółkowska et al. 2024). The model is being further developed to improve pollen and nectar modelling and pesticide simulations, supported by several European/Danish projects, to develop tools for system-based environmental risk assessments for pesticides. Three examples of the application of ALMaSS simulation relying on the landscape model are described below.

A) Policy evaluation (Ziółkowska et al. 2022). The ALMaSS landscape simulation was used to evaluate the effectiveness of policy-driven mitigation measures on carabid beetle populations in Dutch agroecosystems. Three simulation scenarios were created using ALMaSS to simulate mitigation measures, including 1) high, medium, and low toxicity levels of insecticides; 2) different levels of spray drift reduction (50% and 90%); 3) the introduction of 4-meter-wide grassy field margins in 50% of the fields. The beetle species simulation was run on top of these scenarios. The study found that reducing the toxicity of insecticides had the most significant positive impact on beetle populations. Further, the effectiveness of mitigation measures was highly context-specific, influenced by landscape heterogeneity and farming practices.
B) Multi-criteria decision-making to support agri-environment schemes (AES) (Topping et al. 2019). The effectiveness of the ALMaSS model in evaluating typical AES management practices was achieved by combining species models and landscape models. Through the landscape model, thirteen different management scenarios were tested, demonstrating that landscape context plays a crucial role in the effectiveness of agri-environment schemes. The flexibility and adaptability of ALMaSS landscape models can support the assessment and continuous refinement of AES measures.

C) Evaluating consequence of land-use strategies on wildlife populations. This is a typical application for the landscape model. For example, in Jepsen et al. (2005), the impact of converting agricultural land to forest or permanent grass set-aside to increase groundwater protection was evaluated for several species. Positive impacts were observed for carabid beetles and mammal species, while negative impacts were noted for the skylark and spider. Another scenario tested the effect of reducing pesticides by changing the crop composition and management practices. The reduced pesticide mortality led to an increase in carabid beetle, spider, and field vole populations. The results demonstrate the species context dependency of the impacts of the same management.

In all three cases, the scenarios evaluated were driven by changes in the configuration of the ALMaSS landscape model.

Access to and use of the software

ALMaSS is intended for use by researchers to develop their own simulations. It is not an out-of-the-box simulation program. The code for the landscape model can be accessed through a GitLab project ALMaSS_Methodology (https://gitlab.com/ALMaSS/almass_methodology). This is a complete implementation of the landscape model. But, please note that this is only a running example, as it includes only one artificial farm type and uses only two crops in its crop rotation file (UserDefinedFarm27.rot). The source codes for two Danish crops, spring barley and winter wheat, are included as examples. Procedures to compile the code for Linux are provided in the GitLab project.

The project also includes a run directory, which contains an executable version of the code for Linux and the required input files for ALMaSS simulations. If you choose to compile the code yourself, you need to copy the executable to the run directory. By default, it will run a 1-year simulation for the landscape model. In the “Image” subdirectory under the run directory, you can find several images stored from the simulation showing biomass, vegetation type, landscape element type, soil type, and farm ownership for the fifteenth day of each month, after the simulation is run. Different colours represent either different types or different amounts, such as biomass. As shown in Fig. 3 for biomass

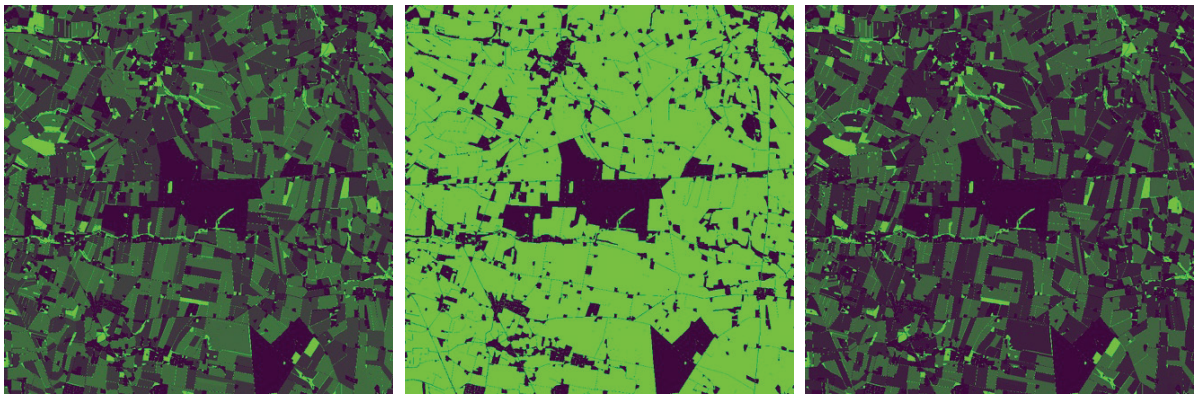

Figure 3. Biomass dynamics in January, July, and December.

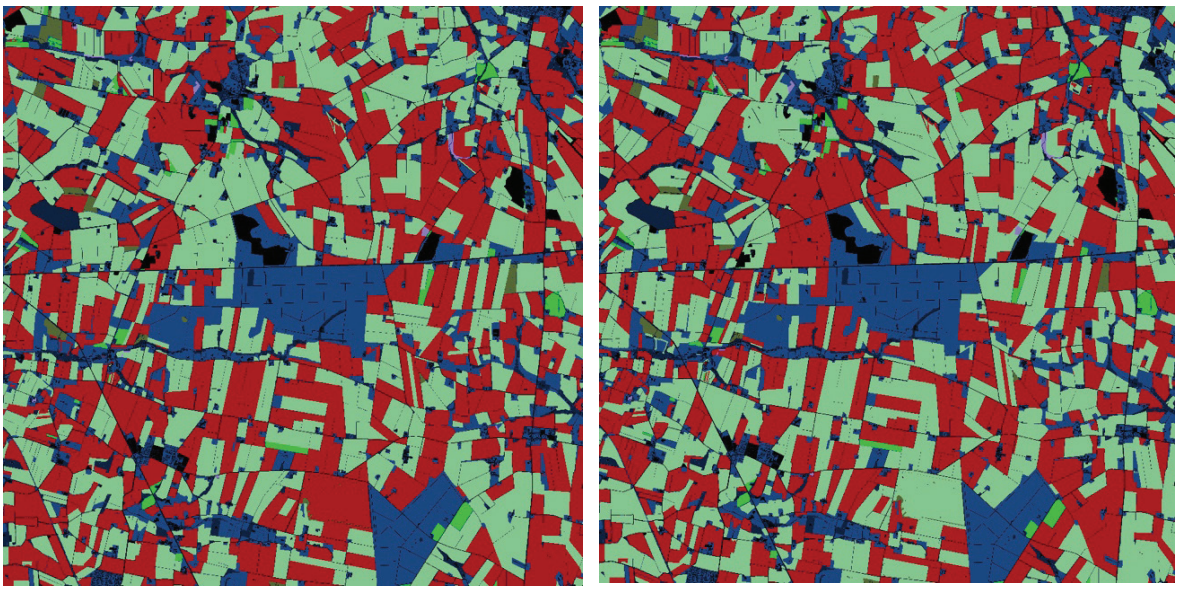

Figure 4. Vegetation types in January and September (winter wheat is represented by pale green while spring barley is displayed as dark red).

dynamics, in January, the crops had little biomass (dark green); in July, they grew to have more biomass (bright green); and by December, the biomass had disappeared (dark green).

The vegetation types in January and September are displayed in Fig. 4 Vegetation types in January and September (winter wheat is represented by pale green while spring barley is displayed as dark red)4. In January, some farm fields had winter wheat (pale green) while others had spring barley (dark red). By September, these fields were switched to the other crop because, in this example, we only have one farm type, and they grow two crops in rotation.

The ALMaSS landscape simulation is flexible. Different scenarios can be set by changing parameters in the "TIALMaSSConfig.cfg" file. For example, removing the # symbols for the lines below the "TIALMaSSConfig.cfg" will create a monocrop scenario. In this scenario, all the farms will only grow winter wheat as shown in Fig. 5 Vegetation type in January and September under a mono-crop scenario using winter wheat (pale green in the figures)5.

#FARM_FIXED_CROP_ENABLE (bool) = true

#FARM_FIXED_CROP_TYPE (int) = 1026

Another important aspect is the use of the ALMaSS simulation as a dynamic context for species simulations. The same GitLab project currently provides three artificial species models and a Skylark model, which users can run as examples. Details about these model types can be found in Topping and Duan (2024), Duan and Topping (2024), Topping and Odderskaer (2004), and Topping et al. (2013).

Suppl. material 1 contains the landscape and associated class documentation generated by doxygen (www.doxy.org). Suppl. material 1 is also available at https://almass.gitlab.io/almass_methodology/Landscape.pdf.

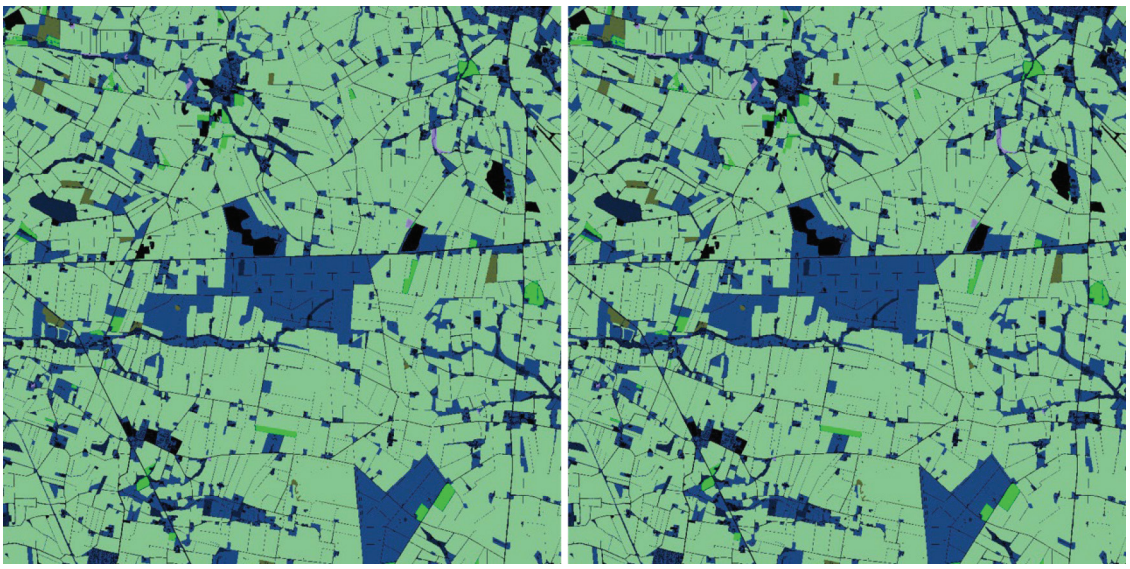

Figure 5. Vegetation type in January and September under a mono-crop scenario using winter wheat (pale green in the figures).

Suppl. material 2 is the original vegetation growth model (Topping and Olesen 2005), reproduced here with some minor corrections to provide a secure online location.

Acknowledgements

Many people have contributed to the ALMaSS landscape model over the years. Martin Rehder and Frank Nikolaisen were instrumental in the early design of the model. Geoff Groom, Luna Kondrup Marcussen and Elżbieta Ziółkowska all have made important contributions to aspects of the model as it is today. This work was partially supported by DeiC National HPC (UCloud, GenomeDK, LUMI) (g.a. DeiC-AU-N1-000025, DeiC-AU-L5-0011 and DeiC-AU-N2-2023015) and by the EcoStack project funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 773554.

Additional information

Conflict of interest

The authors have declared that no competing interests exist.

Ethical statement

No ethical statement was reported.

Funding

This work was partially supported by DeiC National HPC (UCloud, GenomeDK, LUMI) (g.a. DeiC-AU-N1-000025, DeiC-AU-L5-0011 and DeiC-AU-N2-2023015) and by the EcoStack project funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 773554.

Author contributions

Conceptualization: CJT. Methodology: XD. Software: XD, CJT. Writing - original draft: CJT. Writing - review and editing: XD, CJT.

Author ORCIDs

Christopher John Topping https://orcid.org/0000-0003-0874-7603
Xiaodong Duan https://orcid.org/0000-0003-2345-4155

Data availability

All of the data that support the findings of this study are available in the main text or Supplementary Information.

References

Assante M, Boizet A, Candela L, Castelli D, Cirillo R, Coro G, Fernández E, Filter M, Frosini L, Georgiev T, Kakaletris G, Katsivelis P, Knapen R, Lelii L, Lokers RM, Mangiacrapa F, Manouselis N, Pagano P, Panichi G, Penev L, Sinibaldi F (2021) Realising virtual research environments for the agri-food community: The AGINFRA PLUS experience. Concurrency and Computation: Practice and Experience 33: e6087. https://doi.org/10.1002/cpe.6087

Chandra R (2001) Parallel programming in OpenMP. Morgan kaufmann, 240 pp.

Cullum C, Rogers KH, Brierley G, Witkowski ETF (2016) Ecological classification and mapping for landscape management and science: Foundations for the description of patterns and processes. Progress in Physical Geography: Earth and Environment 40: 38–65. https://doi.org/10.1177/0309133315611573

Duan X, Topping CJ (2024) A General Subpopulation Model for the Animal Landscape and Man Simulation System (ALMaSS). Food and Ecological Systems Modelling Journal (in review).

Gamma E, Helm R, Johnson R, Vlissides J (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 195 pp.

Groeneveld D, Tekinerdogan B, Garousi V, Catal C (2021) A domain-specific language framework for farm management information systems in precision agriculture. Precision Agriculture 22: 1067–1106. https://doi.org/10.1007/s11119-020-09770-y

Hoye TT, Skov F, Topping CJ (2012) Interpreting outputs of agent-based models using abundance-occupancy relationships. Ecological Indicators 20: 221–227. https://doi.org/10.1016/j.ecolind.2012.01.017

Janssen MA, Pritchard C, Lee A (2020) On code sharing and model documentation of published individual and agent-based models. Environmental Modelling & Software 134: 104873. https://doi.org/10.1016/j.envsoft.2020.104873

Langhammer M, Thober J, Lange M, Frank K, Grimm V (2019) Agricultural landscape generators for simulation models: A review of existing solutions and an outline of future directions. Ecological Modelling 393: 135–151. https://doi.org/10.1016/j.ecolmodel.2018.12.010

Malawska A, Topping CJ (2016) Evaluating the role of behavioral factors and practical constraints in the performance of an agent-based model of farmer decision making. Agricultural Systems 143: 136–146. https://doi.org/10.1016/j.agsy.2015.12.014

Malawska A, Topping CJ (2018) Applying a biocomplexity approach to modelling farmer decision-making and land use impacts on wildlife. Journal of Applied Ecology 55: 1445–1455. https://doi.org/10.1111/1365-2664.13024

Malawska A, Topping CJ, Nielsen HO (2014) Why do we need to integrate farmer decision making and wildlife models for policy evaluation? Land Use Policy 38: 732–740. https://doi.org/10.1016/j.landusepol.2013.10.025

Minar N, Burkhart R, Langton C, Askenazi M (1996) The swarm simulation system: A toolkit for building multi-agent simulations.

Poulsen T, Duan X, Topping CJ (2023) Modelling dynamic pesticide amounts in multiple environmental compartments at landscape scales in ALMaSS. Food and Ecological Systems Modelling Journal 4: 107849. https://doi.org/10.3897/fmj.4.107849

Terribile F, Acutis M, Agrillo A, Anzalone E, Azam-Ali S, Bancheri M, Baumann P, Birli B, Bonfante A, Botta M, Cavaliere F, Colandrea M, D'Antonio A, De Mascellis R, De Michele C, De Paoli G, Della Monica C, Di Leginio M, Ferlan M, Ferraro G, Florea A, Hermann T, Hoenig H, Jahanshiri E, Jevsenak J, Kárpáti V, Langella G, Le QB, Lezzi D, Loishandl H, Loudin S, Manna P, Marano G, Marotta L, Merticariu V, Mileti FA, Minieri L, Misev D, Montanarella L, Munafò M, Neuwirth M, Orefice N, Pácsonyi I, Panagos P, Perego A, Huu BP, Pinto F, Prebeck K, Puig A, Pump J, Schillaci C, Simoncic P, Skudnik M, Stankovics P, Tóth G, Tramberend P, Vingiani S, Vuolo F, Zucca C, Basile A (2024) The LANDSUPPORT geospatial decision support system (S-DSS) vision: Operational tools to implement sustainability policies in land planning and management. Land Degradation & Development 35: 813–834. https://doi.org/10.1002/ldr.4954

Topping CJ (2022) The Animal Landscape and Man Simulation System (ALMaSS): a history, design, and philosophy. Research Ideas and Outcomes 8: e89919. https://doi.org/10.3897/rio.8.e89919

Topping CJ, Hansen TS, Jensen TS, Jepsen JU, Nikolajsen F, Odderskaer P (2003) ALMaSS, an agent-based model for animals in temperate European landscapes. Ecological Modelling 167: 65–82. https://doi.org/10.1016/S0304-3800(03)00173-X

Topping C, Odderskaer P (2004) Modeling the influence of temporal and spatial factors on the assessment of impacts of pesticides on skylarks. Environmental Toxicology and Chemistry 23: 509–520. https://doi.org/10.1897/02-524a

Topping C, Olesen J (2005) Vegetation growth simulation in ALMaSS 4.0. https://projects.au.dk/fileadmin/ALMaSSVegetationModelling.pdf

Topping CJ, Hoye TT, Olesen CR (2010) Opening the black box-Development, testing and documentation of a mechanistically rich agent-based model. Ecological Modelling 221: 245–255. https://doi.org/10.1016/j.ecolmodel.2009.09.014

Topping CJ, Lagisz M (2012) Spatial dynamic factors affecting population-level risk assessment for a terrestrial arthropod: An agent-based modeling approach. Human and Ecological Risk Assessment 18: 168–180. https://doi.org/10.1080/10807039.2012.632292

Topping CJ, Odderskaer P, Kahlert J (2013) Modelling skylarks (*Alauda arvensis*) to predict impacts of changes in land management and policy: Development and testing of an agent-based model. PLOS ONE 8(6): e65803. https://doi.org/10.1371/journal.pone.0065803

Topping CJ, Alroe HF, Farrell KN, Grimm V (2015) Per aspera ad astra: Through complex population modeling to predictive theory. American Naturalist 186: 669–674. https://doi.org/10.1086/683181

Topping CJ, Luttik R (2017) Simulation to aid in interpreting biological relevance and setting of population-level protection goals for risk assessment of pesticides. Regulatory Toxicology and Pharmacology 89: 40–49. https://doi.org/10.1016/j.yrtph.2017.07.011

Topping CJ, Dalby L, Skov F (2018) Developing spatio-temporal models for landscape-scale pesticide ERA. Copenhagen, Denmark, 146 pp.

Topping CJ, Dalby L, Valdez JW (2019) Landscape-scale simulations as a tool in multi-criteria decision making to support agri-environment schemes. Agricultural Systems 176: 102671. https://doi.org/10.1016/j.agsy.2019.102671

Topping CJ, Duan X (2024) Managing large and complex population operations with agent based models: The ALMaSS Population_Manager. Food and Ecological Systems Modelling Journal 5: 117593. https://doi.org/10.3897/fmj.5.117593

Topping CJ, Bednarska AJ, Benfenati E, Chetcuti J, Delso N, Duan X, Focks A, Laskowski R, Lombardo A, Marcussen L, Metodiev T, Rubinigg M, Rundlöf M, Sgolastra F, Stoyanova C, Sušanj G, Williams J, Ziółkowska EM (2024) PollinERA: Understanding pesticide-pollinator interactions to support EU environmental risk assessment and policy. Research Ideas and Outcomes 10: e127485. https://doi.org/10.3897/rio.10.e127485

Treude C, Middleton J, Atapattu T (2020) Beyond accuracy: Assessing software documentation quality. In: 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE). Electr Network, 1509–1512. https://doi.org/10.1145/3368089.3417045

Ziółkowska E, Topping CJ, Bednarska AJ, Laskowski R (2021) Supporting non-target arthropods in agroecosystems: Modelling effects of insecticides and landscape structure on carabids in agricultural landscapes. Science of the Total Environment 774: 145746. https://doi.org/10.1016/j.scitotenv.2021.145746

Ziólkowska E, Tiktak A, Topping CJ (2022) Is the effectiveness of policy-driven mitigation measures on carabid populations driven by landscape and farmland heterogeneity? Applying a modelling approach in the Dutch agroecosystems. PLOS ONE 17: e0279639. https://doi.org/10.1371/journal.pone.0279639

Ziółkowska E, Laskowski R, Dominic AR, Stein S, Wenzel B, Kehlenbeck H, Riddervold M, Topping CJ (2024) Evaluating the impact of landscape structure and source-sink dynamics on non-target arthropod pesticide risk assessments in Germany. Umweltbundesamt, Dessau-Roßlau, Germany, 150 pp. https://www.umweltbundesamt.de/sites/default/files/medien/11850/publikationen/58_2024_texte_elonta.pdf

Supplementary material 1

The code documentation for the ALMaSS Landscape Model

Authors: Christopher John Topping, Xiaodong Duan

Data type: pdf

Explanation note: Landscape and associated class documentation generated by doxygen (www.doxy.org).

Copyright notice: This dataset is made available under the Open Database License (http://opendatacommons.org/licenses/odbl/1.0/). The Open Database License (ODbL) is a license agreement intended to allow users to freely share, modify, and use this Dataset while maintaining this same freedom for others, provided that the original source and author(s) are credited.

Link: https://doi.org/https://doi.org/10.3897/fmj.5.121215.suppl1

Supplementary material 2

Vegetation growth simulation in ALMaSS

Authors: Christopher John Topping, Xiaodong Duan

Data type: pdf

Explanation note: Vegetation growth model (Topping and Olesen 2005), reproduced here with some minor corrections to provide a secure online location.

Copyright notice: This dataset is made available under the Open Database License (http://opendatacommons.org/licenses/odbl/1.0/). The Open Database License (ODbL) is a license agreement intended to allow users to freely share, modify, and use this Dataset while maintaining this same freedom for others, provided that the original source and author(s) are credited.

Link: https://doi.org/https://doi.org/10.3897/fmj.5.121215.suppl2