The LUE data model

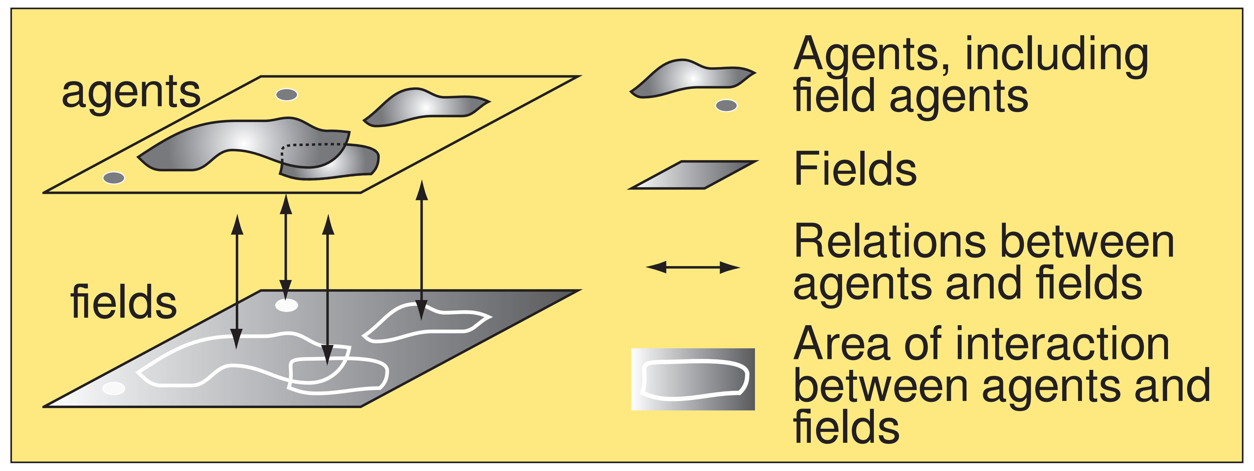
for agents and fields

Kor de Jong (k.dejong1@uu.nl), Oliver Schmitz, and Derek Karssenberg - Department of Physical Geography, Faculty of Geosciences, Utrecht University, Utrecht, The Netherlands

The LUE data model makes it convenient to write efficient environmental modeling software in which both agents and fields are manipulated

Data models are used in software to describe how information is organized. Popular used data models in environmental modeling software include the raster data model for representing continuous varying spatial information, and the vector data model for representing discrete objects located in space. In agent-based modeling, agents are often represented by a general object data model, where the modeler has to define which properties make up a certain class of agents.

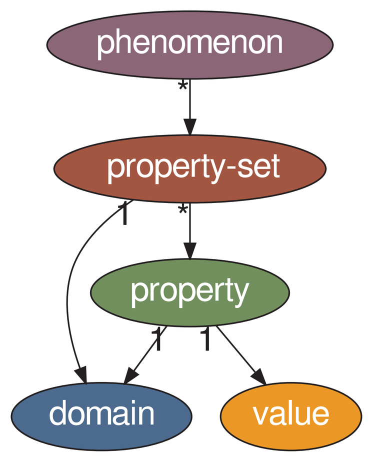
We work on the LUE conceptual data model for representing agent- and field-based data, and its current implementation in a physical data model (a dataset format). The intended audience of LUE is developers of environmental modeling software.



Requirements

✓ Capable of representing all kinds of data used in environmental modeling, e.g.:
  ✓ Data located in time and/or (3D) space
  ✓ Data that varies continuously through time and/or space
  ✓ Data that varies spatial location through time
  ✓ Linked data: networks, relations
✓ Allow for an efficient implementation (locality of reference, support for parallel I/O)

Conceptual data model



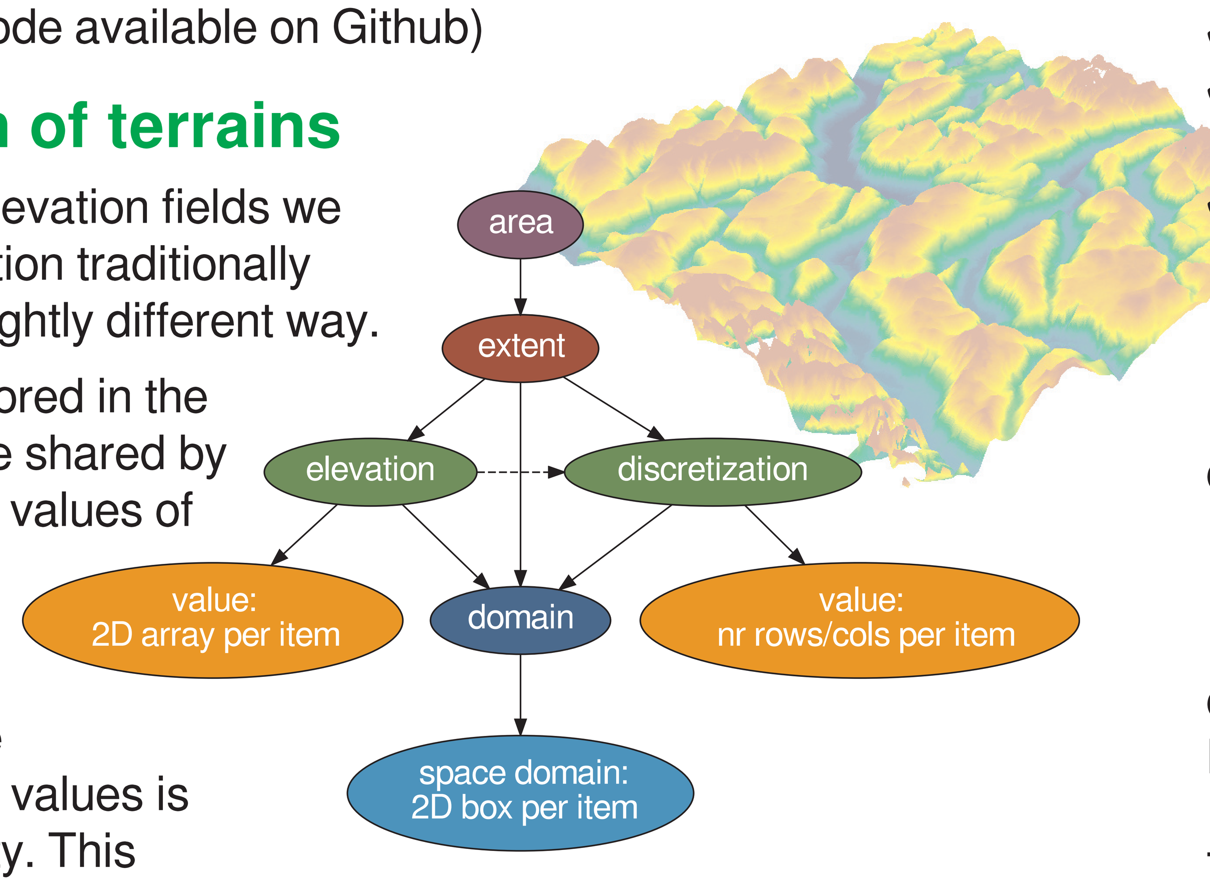
Phenomenon: Collection of related property-sets, e.g.: properties of individual birds
Property-set: Collection of properties sharing a time/space domain
Domain: Information about when and where something 'is', e.g.: location through time of individual birds
Property: Location and variation of a characteristic through time and space
Value: Magnitude of a property, e.g.: speed of individual birds
Item: Identifies an individual/object/agent

The domain and value contain information for all items in the property-set.

Physical data model

Implementation of concrete conceptual data models in HDF5. Some characteristics:
✓ All model data in a single, portable file
✓ Support parallel I/O
✓ C++ API (C++14) and Python API (with support for Numpy arrays)
✓ Open source software (code available on Github)

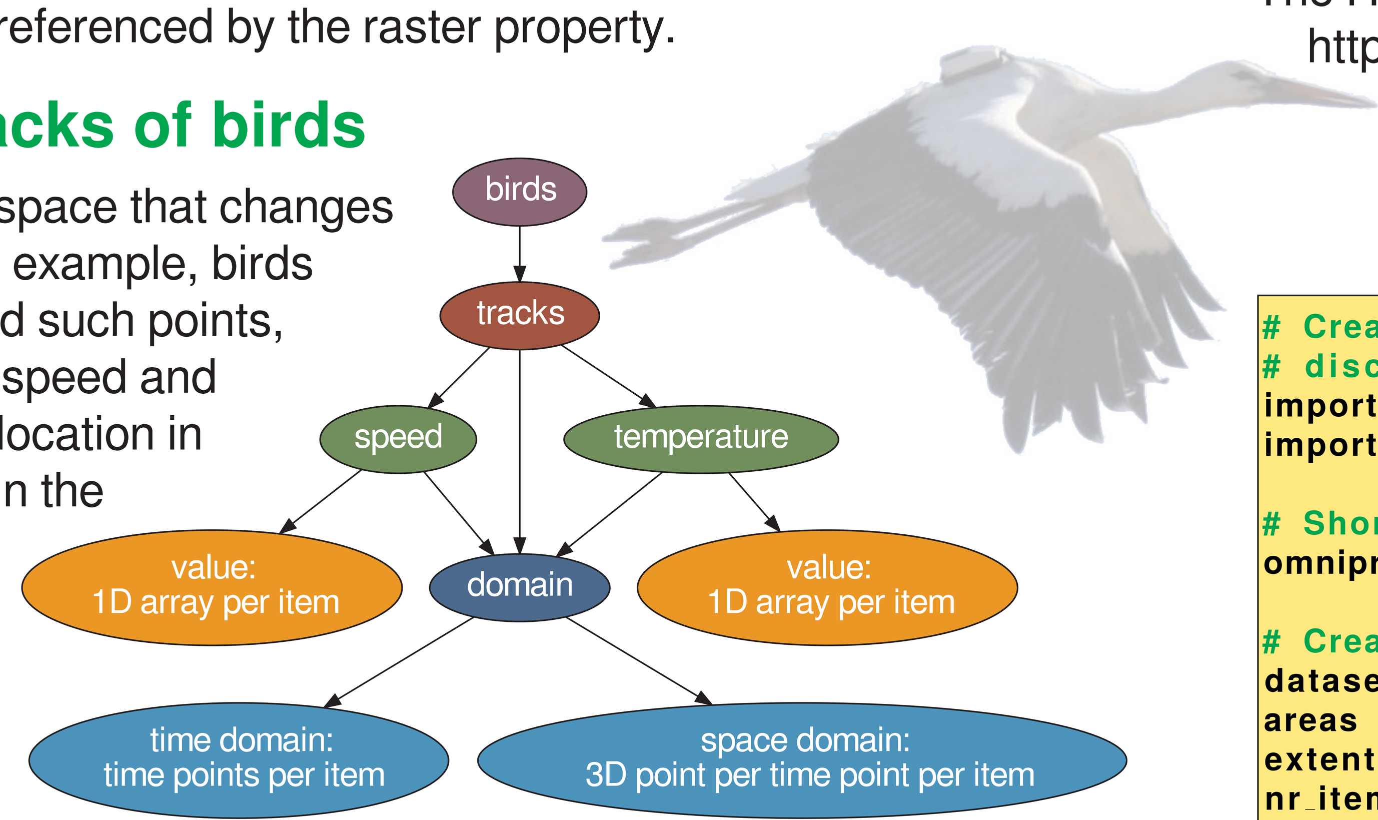
Example: Elevation of terrains



To represent one or more elevation fields we can store the same information traditionally stored in rasters, but in a slightly different way.

The extent of the fields is stored in the domain. This domain can be shared by multiple properties. The cell values of each field are stored in the value. The property aggregates the domain and value. Information about the discretization of each field's values is stored in a separate property. This discretization property is referenced by the raster property.

Example: GPS tracks of birds



A GPS track is a point in space that changes location through time. For example, birds carrying a GPS can record such points, along with properties like speed and temperature. In LUE, the location in time and space is stored in the domain. All properties that are recorded by the same GPS device share the same domain, so are stored in the same property-set.

Current status

The implementation of the LUE dataset format is work in progress. We are focusing on the following initial set of concrete conceptual data models:
✓ Constant size of item collection
  ✓ Scalars
  ✓ Rasters
  ✓ Stationary temporal rasters
  ✓ Mobile points (space-time paths, GPS tracks)

Opportunities

Given the LUE data model the following becomes possible (examples):
✓ Representing temporal varying discretization. The change in discretization can happen at different time points than the change in property value.
✓ Storing fields of different resolution in the same property-set. This can be convenient to speed up certain parts of the model.
✓ Storing and accessing field and agent data in a uniform way.
✓ For the same phenomenon (with the same set of items), we can now store properties with very different domains.
✓ A single high level API for data that is organized very differently internally. The details are stored where they belong: in the lower levels of the software stack.

More info

de Bakker, M.P., de Jong, K., Schmitz, O., Karssenberg, D., 2016. *Design and demonstration of a data model to integrate agent-based and field-based modeling*. Environmental Modelling and Software. http://dx.doi.org/10.1016/j.envsoft.2016.11.016
de Jong, K., 2017. *LUE source code*. https://github.com/pcraster/lue
PCRaster R&D Team, 2000-2017. *PCRaster Environmental Modelling Software*, http://www.pcraster.eu
The HDF Group, 2000-2017. *Hierarchical data format version 5*, http://www.hdfgroup.org/HDF5

```
# Create a new dataset and write a number of fields as rasters with random
# discretizations and random cell values.
import numpy
import lue

# Shortcut to sub-module
omnipresent = lue.constant_size.time.omnipresent

# Create dataset
dataset = lue.create_dataset("some_project.lue")
areas = dataset.add_phenomenon("areas")
extents = omnipresent.create_property_set(areas, "extents")
nr_items = 100

# Add a discretization property containing nr_rows/nr_cols for each raster
value_shape = (2, )
value_type = numpy.uint32
discretization = omnipresent.same_shape.create_property(extents, "discretization",
    value_type, value_shape)
raster_shapes = numpy.arange(start=1, stop=nr_items * 2 + 1, dtype=value_type) \
    .reshape((nr_items, 2))
nr_cells = discretization.reserve(nr_items)
nr_cells[:] = raster_shapes

# Add a property containing the raster cell values for each raster
rank = 2
value_type = numpy.int32
elevation = omnipresent.different_shape.create_property(extents, "elevation",
    value_type, rank)
values = elevation.reserve(nr_items)
for i in range(nr_items):
    values[i][:] = (10 * numpy.random.rand(*raster_shapes[i])).astype(value_type)

# Link discretization to property
elevation.discretize_space(discretization)
```