Environmental Modelling & Software 89 (2017) 172–189

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Design and demonstration of a data model to integrate agent-based and field-based modelling

Merijn P. de Bakker*, Kor de Jong, Oliver Schmitz, Derek Karssenberg

Department of Physical Geography, Faculty of Geosciences, Utrecht University, Heidelberglaan 2, PO Box 80115, 3508 TC Utrecht, The Netherlands

ARTICLE INFO

Article history:
Received 2 September 2016
Received in revised form
27 September 2016
Accepted 12 November 2016
Available online 12 January 2017

Keywords:
Agent-based modelling
Field-based modelling
Integrated modelling
Data model
Map algebra
Modelling language

ABSTRACT

Dynamic environmental modelling of spatio-temporal systems often requires the representation of both fields and agents. Fields are continuous with values in the whole spatio-temporal domain of a model, while agents are bounded in space and often mobile. It is currently difficult for environmental modellers with limited software engineering background to construct such field-agent models, as modelling frameworks mostly do not support the integration of fields and agents. To overcome this issue, we describe a data model combining fields and agents in a single concept. This data model represents fields, agents and relations by grouping items sharing properties into a phenomenon. The concepts domain, property set and value handle spatio-temporal attribute representations. The data model is implemented in a software prototype that shows how data on fields and agents is stored and manipulated.

1. Introduction

Dynamic environmental models represent the temporal evolution of the environment by applying a state transition function on the system's state variables, mostly at each time step or for a series of events occurring over time. Their state variables often vary in the spatial domain as well as in the temporal domain, which requires that an appropriate representation of modelled phenomena has to be used that is capable of storing spatio-temporal information. Two approaches currently exist (Filatova et al., 2013; Goodchild, 2013; Parker, 2005), as shown in Fig. 1. In the field-based approach, attributes are assumed to have a value defined everywhere in space-time, often discretized in cells or voxels and time steps. Examples of field-based approaches are aboveground biomass represented by a raster map, where each grid cell value represents the biomass in the grid cell, for each time step, or atmospheric pressure, where each voxel is assigned a pressure value. In the agent-based approach, attributes are assigned to, sometimes mobile, entities bounded in space-time. Agents are sometimes referred to as individuals in ecology and objects in geographical information science (e.g. Grimm and Railsback, 2013; Goodchild et al., 2007). An example of the agent-based approach is where the modeller chooses to represent biomass as a biomass value assigned to each individual tree, in order to simulate, for instance, tree growth of individual trees (e.g. Valentine and Mäkelä, 2005).

In a large number of environmental problems, in particular those that require an integrated approach, there is a strong need to use environmental models that combine the agent-based and field-based approaches. This is because many systems contain both phenomena that are better represented by fields and those that are better represented by agents. In addition, there is often interaction between fields and agents (Fig. 1). In ecosystem models, for instance, animals are often best represented as agents, while their habitat is better represented as a field (e.g. Schippers et al., 2014; Bennett and Tang, 2006). Other examples include interactions between water users (agents) and groundwater dynamics (field) (Castilla-Rho et al., 2015), the influence of weather (fields) on trees (agents) (Schelhaas et al., 2007), and modelling of air pollution (field) and personal exposure to this air pollution of mobile individuals (Sbihi et al., 2015).

The integration of fields and agents in models is however still complicated from the perspective of model implementation, as standardized software is lacking. Generally speaking, field-agent integration can be approached in two ways. One is to build models from scratch with a general-purpose language (Fig. 2A), for instance FORTRAN or C++. With such a language, the modeller has

* Corresponding author.
E-mail address: m.p.debakker@uu.nl (M.P. de Bakker).

http://dx.doi.org/10.1016/j.envsoft.2016.11.016
1364-8152/

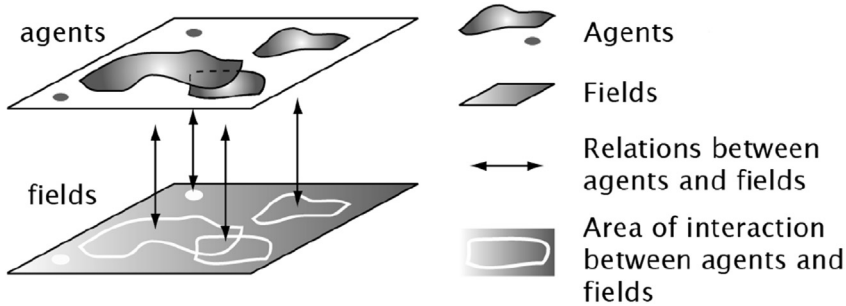


Fig. 1. An integrated agent-based and field-based model consists of *agents*, which are discrete entities in space (points, lines, areas) and are possibly mobile, and *fields*. Fields represent continuous attributes. Many agents have continuous variation within their spatio-temporal extent. Agents and fields interact depending on location and extent of agents.

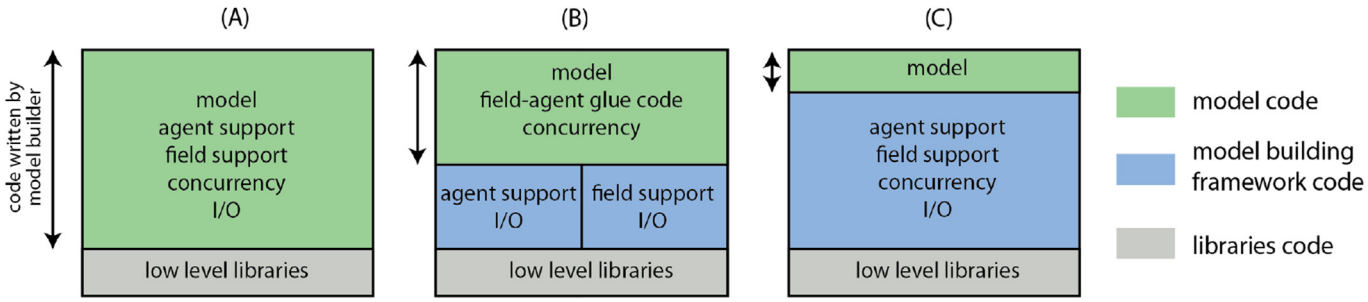


Fig. 2. Share of model code, framework code and general-purpose programming libraries code in three different approaches to programming models. (A), Using generic, general-purpose programming libraries to build a model from scratch; (B), Using separate model building frameworks for agents and fields; (C), proposed approach using a model building framework that supports fields and agents. Of the approaches presented in the figure, approach (C) requires the least programming effort from the modeller.

to implement software using elementary constructs which requires extensive software engineering skills, especially if the model implementation is challenging, e.g. when the model needs to be executed on a supercomputer. With a few exceptions in certain research areas this exceeds the programming capabilities of most domain specialists. Using these languages comes with a number of disadvantages (Karssenberg, 2002; van Engelen, 2001) and shifts the modeller's focus from describing domain processes to implementing numerical algorithms. A solution to this problem is offering model builders specialized programming interfaces tailored to their domain of knowledge (Fig. 2B). Domain experts then program models using a model building software framework (e.g. Karssenberg et al., 2010), which provides a programming interface matching the conceptual level of thinking of domain specialists. The building blocks of models can be offered in a generic scripting language or, in its most sophisticated form, as a Domain-Specific Language (DSL) (Mernik et al., 2005; van Deursen et al., 2000), which is a language that matches the semantics of a specific domain of use, improving productivity, verification and optimization (Kosar et al., 2010; Fisher, 1999). This approach of using model building software frameworks is promising because it becomes easier for domain specialists to program models (Holst and Belete, 2015). However, the current key limitation of model building software frameworks is that they focus on either field-based (e.g. Karssenberg et al., 2010) or agent-based modelling (e.g. NetLogo, 2012), but not both (Carneiro et al., 2013). This is inconvenient because coupling involves additional technical overhead and thus makes it more difficult for domain experts to build models.

The disadvantages of the current situation would be resolved by a model building framework supporting both fields and agents (Fig. 2C), which preferably includes a domain-specific language capable of manipulating fields and agents. The design of such a language is mostly based on a semantic model of the application domain (e.g. Fowler, 2010), e.g a (conceptual) data model. In this paper, we aim at the development and software implementation of a conceptual and physical data model integrating fields and agents that is useful in a domain-specific language by supporting multi-paradigm operations. The development of a new data model is required as existing data models, with respect to field-agent integration in dynamic environmental models, are still lacking regarding a number of aspects. Recent important contributions have approached integration of fields and agents from a theoretical point of view (e.g. Goodchild et al., 2007; Kjenstad, 2006; Galton, 2001), but the use of these concepts in a framework for dynamic environmental modelling is thus far underdeveloped. Also, existing physical, implemented, data models used in environmental model building frameworks (e.g. North et al., 2013; Karssenberg et al., 2010) do not integrate fields and agents. Further, to our knowledge, concepts and generic software frameworks that explicitly deal with spatio-temporal attribute variation within the extent of agents do currently not exist. This is relevant in models that model the evolution of individual features in a landscape, including their internal variation, for instance ecosystem models simulating temporal changes in the heterogeneity of foliage in individual tree crowns (Valentine and Mäkelä, 2005).

A key requirement of the conceptual data model is that it needs to function as an input argument of functions in a domain-specific language for dynamic environmental modelling. We build upon the concept of map algebra, introduced by Tomlin (1983) and since then used in many model building software packages for manipulation of fields (e.g., ESRI, 2015; van Deursen et al., 2000). Map algebra offers syntactically simple statements in the style of algebraic expressions on rasters instead of numbers. In our envisioned modelling language, the input arguments, which are raster maps in

the case of map algebra, could also be agents. This implies that at the highest level, fields and agents need to have a common representation in our conceptual data model.

The key question that needs to be answered is how agents and fields can be conceptualized such that these can be represented by a single conceptual data model that can function as input argument in a map-algebra like environmental modelling language. In the following section we therefore specify requirements of the conceptual data model and review existing conceptual and physical data models. Section 3 describes the conceptual data model and in Section 4 we describe how we address the posed requirements. We also provide a prototype software implementation of the data model based on HDF5, using an example of a vegetation grazing model that integrates field-based and agent-based modelling (Section 5). Section 6 discusses the approach and results of this paper.

2. Requirements of the data model and previous work

The conceptual data model that is being developed is required to represent fields, agents (including agents with internal variation) and relations between these, i.e. networks. Also, it needs to integrate these types and it needs to be applicable in a map algebra like-language. This section discusses these requirements and existing concepts that we can build upon in the development of the conceptual data model. Table 1 provides an overview.

2.1. Fields

Field-based modelling is defined as approaches that give a spatially and temporally continuous representation of attributes in the environment, which implies that values of an attribute are in principle modelled everywhere in 2D or 3D space and time (Karssenberg et al., 2010; Galton, 2004; Peuquet, 2001). In modelling practice, a field is typically bounded in space and time, for example by the study area. Examples of attributes represented as fields are air temperature in weather forecast models (Molteni et al., 1996) and topographical elevation in hydrological models (Sutanudjaja et al., 2011) because these phenomena have a value defined everywhere in space and time. When modelling continuous fields, the modeller translates the continuous attribute into a physical data model by discretization, in order to apply numerical algorithms. Widely used spatial discretizations are rasters and TINs (Kjenstad, 2006; Winter, 1998). Several programming environments for field-based modelling exist, among which are MATLAB (2012), a general purpose modelling environment, and PCRaster (Karssenberg et al., 2010; van Deursen, 1995) for spatio-temporal modelling.

2.2. Agents

Next to field-based modelling, modellers can choose to model their system using discrete entities, or objects. The main characteristic of objects as opposed to fields is that they are bounded in space. The spatial characteristic also translates to the temporal dimension, which implies that an object exists for a certain period of time (Peuquet, 2001). Table 1 provides examples of agent-based models.

From a modelling perspective, objects are often referred to differently. In ecology, the term individual-based modelling is used, while in the Geographical Information Science (GIS) community one often prefers the term object-based modelling. Object-based modelling in GIS has traditionally concentrated on spatial modelling only and little attention has been paid to the time domain (Pultar et al., 2010; Brown et al., 2005). This is opposite to individual-based models in ecology which are often dynamic, and are often called agent-based if applied to societal problems (Bithell et al., 2008). In these models, individuals or agents are usually mobile and interact with each other and the environment (e.g. Crooks and Castle, 2012). Hence, we choose to use the concept of 'agent' instead of 'object' to represent all kinds of discrete individuals that are bounded in space since it also includes mobility and interaction, which are both important in many dynamic models. This deviates from the traditional use of 'object' as opposed to 'field', which we view as having a subset of the properties of an agent. Many specialized agent-based libraries, frameworks and tools exist (Abdou et al., 2012; Crooks and Castle, 2012). Some are not intended to explicitly model spatial behaviour (e.g. Altreva, 2016), but many allow for explicit spatial modelling, of which GAMA (Grignard et al., 2014), GeoMason (Sullivan et al., 2010), NetLogo (2012) and RePast (North et al., 2013) are notable examples. Field-based modelling in these agent-based frameworks is

Table 1
Overview of *conceptual models*, *software* and *applications* that deal with the specific requirements put forward in the section. The literature that is mentioned shows the emphasis of the offered approach. The columns give a non-exhaustive list of examples.

	Conceptual model	Software	Applications
Fields	Longley et al. (2011), Burrough and McDonnell (1998)	MATLAB (MATLAB, 2012), PCRaster (Karssenberg et al., 2010; van Deursen, 1995), RasDaMan (Baumann et al., 1999)	Hydrological models (Sutanudjaja et al., 2011), air pollution modelling (Hoek et al., 2008)
Agents	Grimm and Railsback (2013)	GAMA (Grignard et al., 2014), RePast (North et al., 2013), TerraME (Carneiro et al., 2013), NetLogo (NetLogo, 2012), Mason/GeoMason (Sullivan et al., 2010)	Economic modelling (Parker and Filatova, 2008), social simulation (Ziervogel et al., 2005)
Agents with internal variation	Clarke (2007) (deltatrons)	*no generic integrated software implementations known*	Cloud cover and solar radiation (Kocifaj and Kómar, 2016), reflection of glaciers (Mernild et al., 2015), air pollution (Jjumba and Dragicevic, 2015), sediment modelling (Karssenberg and Bridge, 2008), tree growth (Valentine and Mäkelä, 2005)
Networks	Galton and Worboys (2005), Goodrich and Tamassia (2002)	graph-tool (Peixoto, 2014), R (R Core Team, 2013), NetworkX (Hagberg et al., 2008)	Farm/household interactions (Berger, 2001)
Integration of fields and agents	Ferreira et al. (2014), OGC (2011), Voudouris (2010), Liu et al. (2008), Moreno et al. (2008), Goodchild et al. (2007), Kjenstad (2006), Cova and Goodchild (2002), Câmara et al. (2000)	TerraME (Carneiro et al., 2013)	Air pollution (Jjumba and Dragicevic, 2015), animals and environment (Schippers et al., 2014), wind and trees (Schelhaas et al., 2007; Bennett and Tang, 2006)
Map algebra language	Frank (2005), Tomlin (1983)	Multidimensional Map Algebra (Mennis, 2010), MapScript (Pullar, 2001)	Vegetation change using Multidimensional Map Algebra (Mennis et al., 2005)

often troublesome. Except for GAMA (Grignard et al., 2014) and NetLogo (2012), agent-based modelling frameworks usually require modellers to use general-purpose object-oriented data models and languages (Crooks and Castle, 2012).

2.3. Agents with internal variation

Many research questions include agents that have a continuous spatial and temporal variation within their spatio-temporal extent. For instance, in a number of air pollution cloud models (Jjumba and Dragicevic, 2015), the movement of clouds is simulated, including spatio-temporal variation of pollutant density within a cloud. Likewise, in vegetation models simulating competition for solar radiation between individual trees (Valentine and Mäkelä, 2005), the biomass of a tree crown may vary within its spatio-temporal extent. Other examples include channel belts (Karssenberg and Bridge, 2008), spatial variation of albedo over glaciers (Mernild et al., 2015) or solar energy models simulating heterogeneous cloud covers and their effect on solar radiation diffusion (e.g. Kocifaj and Kómar 2016). These examples reveal that systems that can be represented as agents often also have properties that vary continuously within the spatio-temporal extent associated with each agent.

Few concepts are designed that model agents with internal variation and to our knowledge there is no conceptual data model that represents such agents, although the Deltatrons introduced by Clarke (2007) are an important contribution and starting point. The same goes for software implementations. To our knowledge no framework supports the modelling of spatio-temporal bounded agents with internal variation. As conceptual models and software for agents with internal variation are lacking, models using this representation are currently programmed from scratch using general-purpose programming languages or programmed by adding components to field-based or agent-based software frameworks (e.g. Jjumba and Dragicevic, 2015; Karssenberg and Bridge, 2008).

2.4. Networks between agents

Agents often are part of a network, e.g. social and physical networks (e.g. Crooks and Castle, 2006; Schelhaas et al., 2007; Parker et al., 2003; Bonabeau, 2002; Berger, 2001). Networks can have temporal variation that is unrelated to the variation in the attributes of the connected agents. For instance, the duration and properties of an interaction event between persons is often different from other temporal attributes of the persons.

Peuquet (2001) and Kuhn (2012) argue that a data model for discrete entities ideally incorporates the representation of networks, whether they are spatially explicit, such as roads and rivers, or implicit, e.g. social networks. Network and graph theory in general is well established, and also in the area of spatial modelling literature deals with networks (e.g. Galton and Worboys, 2005; Goodrich and Tamassia, 2002).

Specialized software exists to model networks, often as libraries designed for particular programming languages, e.g. NetworkX (Hagberg et al., 2008) and graph-tool (Peixoto, 2014). Besides these, other frameworks, such as R (R Core Team, 2013) and Repast (North et al., 2013) have network programming capabilities.

2.5. Useful in a modelling language

In dynamic environmental models, fields and agents often need to be connected (e.g. Schippers et al., 2014; Filatova et al., 2013; Schelhaas et al., 2007; Bennett and Tang, 2006). Fig. 1 schematically shows the ideal situation in which fields and agents (including agents with variation within their extent) are represented and dynamically interact within a system. Consequently, a modelling language is required that facilitates the integrated modelling of systems containing fields and agents.

We envision that *map algebra* (Tomlin, 1990) is a suitable principle as foundation for such a modelling language, because it provides modellers with a technique to manipulate data at a high conceptual level, corresponding to the way modellers think of spatial data. Tomlin (1990) developed map algebra as an algebra to manipulate raster maps, i.e. fields. Currently, map algebra is widely used for manipulating raster maps using algebraic notation. In ESRI ArcGIS ESRI (2015), map algebra is implemented as a calculator in which raster maps can be manipulated and combined. The map algebra concept is also realized in MapScript (Pullar, 2001) and TGRASS (Gebbert and Pebesma, 2014). Likewise, PCRaster (Karssenberg et al., 2010; van Deursen, 1995) includes operations on maps that implicitly iterate over cells in the map. Extensions of the original map algebra have been studied (Mennis, 2010; Wang and Pullar, 2005; Câmara et al., 2005), but to our knowledge not in the domain of agent-based modelling. In the following, we show in a very simple manner how we aim to suit map algebra for modelling both fields and agents.

Consider the map algebra statement:

$$\mathbf{map.c} = \mathbf{map.a} + \mathbf{map.b}$$

In this statement, where a, b and c are variables, two maps are combined into a new map. This is done by applying the operation + to the values of the variable a and b for each cell in the map using implicit iteration. Field-based modelling frameworks, such as PCRaster (Karssenberg et al., 2010; van Deursen, 1995) have implemented this style of map algebra. The underlying data model contains rasters for the maps. If we consider agent-based modelling frameworks, it becomes clear that their data model and high-level modelling perspective is very different. Suppose that one wants to apply the same addition operation to attributes of agents in a particular simulation. In agent-based tools, such as NetLogo (2012), one typically has to program the procedure using explicit iteration over agents:

$$\begin{aligned}&\mathbf{agents} = [\mathbf{agent\ definition\ and\ instantiation}]\\&\mathbf{for\ agent\ in\ agents:}\\&\quad \mathbf{agent.c} = \mathbf{agent.a} + \mathbf{agent.b}\end{aligned}$$

This way of programming at a low level of abstraction is not the most convenient for modellers. Firstly, when an object-oriented approach is used in model development, which is usually the case in agent-based modelling (Abdou et al., 2012), one has to handle the process of object definition, instantiation and management. Secondly, in the part of the model where agents are updated to represent their evolution over time, one has to take care of control flow procedures to iterate over agents. Together, this requires a considerable amount of code which possibly results in a time consuming and error prone development process. Thus, we propose to eliminate the low-level procedures in agent-based modelling, by following an approach similar to map algebra:

$$\mathbf{agents.c} = \mathbf{agents.a} + \mathbf{agents.b}$$

Now, the operation iterates implicitly over all agents in *agents*, adding variable a to b for each agent. Next, we can make the arguments for fields and agents uniform, leading to a field-agent algebra similar to map algebra:

$$\mathbf{phenomenon.c} = \mathbf{phenomenon.a} + \mathbf{phenomenon.b}$$

Now, *phenomenon* can take the place of a field or an agent. To be

able to do this, we need a data model that can represent both fields and agents. A common data model instead of two separate data models comes with clear semantics for the modeller, both when modelling fields and agents. This is an advantage when using a domain-specific language. Further, many phenomena appear to have characteristics of both fields and agents, as will be shown below. In a common data model, we can address this more easily compared to a situation where different data models are used to represent fields and agents.

2.6. Integration of fields and agents in a data model

The preceding section has shown how we envision the integrated modelling of fields and agents. It was argued that a common data model for fields and agents is required to reach this objective.

Much work has been done on the design or improvement of integrated representations from theoretical and implementation oriented perspectives. Liu et al. (2008) introduce the 'General-Field' as a generalization of objects and object-fields. General-fields are constructed by mappings from fields to object-fields to objects. Geo-atoms and geo-dipoles are introduced in Goodchild et al. (2007). Moreno et al. (2008) present the VecGCA model, replacing raster-based cellular automata by a set of geometrically evolving objects (polygons) in order to dismiss the sensitivity of cellular automata to scale. Transition rules determine how the objects transform during a simulation, supported by the vectorization and rasterization between rasters and objects. Data models proposing an integration of objects and fields have been described in Câmara et al. (2000), Kjenstad (2006) and Voudouris (2010). Câmara et al. (2000) presents the *geographical object*, under which simple, composite, homogeneous and non-homogeneous geo-objects are classified. A minimal set of common GIS operations on these types of geographical objects is introduced. The PGOmodel (Kjenstad, 2006) is introduced as a base model for fields and objects in GIS. Kjenstad (2006) conceptualizes fields as a set of functions from a parameter space domain to a value space that is multidimensional instead of a mapping from a geometry domain to a value space. It is not explained how the temporal dimension is accounted for in this model. Voudouris (2010) proposes an extension of Kjenstad (2006) to allow for the modelling of object-fields (Cova and Goodchild, 2002) and uncertainty and semantics. In the *Object-Field Model* reflective *phenomena* are linked to *Elementary_geoParticles*, which are central in the model. The *Elementary_geoParticle* itself does not have predefined geometry, but gets its definition from attributeset, behaviourset, semantic and uncertainty classes. Ferreira et al. (2014) takes a different perspective on spatio-temporal data representation. It is shown by Ferreira et al. (2014) that three data types, that is time series, coverages and trajectory, are sufficient to represent changing objects and events. An algebra on these data types allows for the manipulation of spatio-temporal data. However, the model does not represent relations and the presence of different data types still requires specific operations dependent on the data type. Yuan (2001) and Li et al., (2014) approach integration of discrete and continuous dynamic modelling via the representation of events, processes and states.

The Open Geospatial Consortium (OGC) has worked on standardization for geographic data representation and exchange, among which is the OGC General Feature Model (OGC, 2011). Several abstract specifications have been designed (e.g. OGC, 2009; OGC, 2006; OGC, 1999) in order to provide programmers with standards for implementations in which data interoperability and exchange are important (Winter and Nittel, 2003). The OGC specifications for representing geographical data have not yet resulted in an integrated data model that is implemented in a modelling language for heterogeneous systems.

As was argued in the introduction, the distinction between the field-based and agent-based approaches can be found in software used in modelling of environmental systems. Geographic Information Systems are still divided into the raster and vector data models. Also, GIS is currently not well suited for dynamic modelling. Although the urge to generalize the modelling of geographic phenomena is recognized, it is considered unlikely that GIS data models for dynamic modelling and simulation will emerge in the near future (Goodchild, 2013). Specialized frameworks such as PCRaster (Karssenberg et al., 2010), Repast (North et al., 2013), R (R Core Team, 2013) (with the correct modules) and Matlab (MATLAB, 2012) are currently often used for dynamic modelling, but they do not integrate fields and agents within their data model or language. An exception is Carneiro et al. (2013), which brings together field-based and agent-based modelling in a single framework. The urge to store data on both objects and fields is also recognised in the database community. Baumann (1994) identified that conventional DataBase Management Systems (DBMS) are not suited for the storage of raster data. Research in this area of database systems has resulted in the development of array databases such as RasDaMan (Baumann et al., 1999) and SciDB (Brown, 2010) as an alternative for relational databases. However, databases are not well suited for dynamic modelling and simulation.

3. Conceptual data model

In this section we introduce a conceptual data model for the representation of spatio-temporal data as described in the previous section.

3.1. Elements of the conceptual data model

The conceptual data model consists of six basic concepts, namely *phenomenon, item, property, domain, property set* and *value* (Fig. 3);

- The *phenomenon* is the top level concept in the data model. It represents a thing or occurrence in the real world system. A phenomenon contains one or more individuals, called items.
- An *item* is an individual contained in a *phenomenon.*
- A *property* is a particular attribute of a *phenomenon.*
- A *domain* is the area and time period for which a *property* exists, for each *item.*

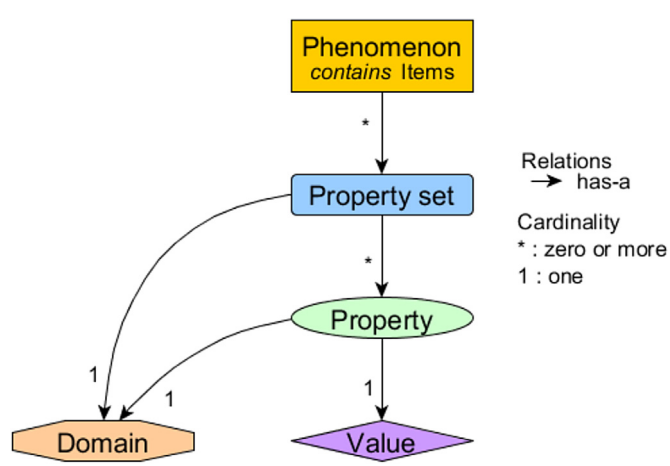


Fig. 3. Conceptual data model for spatio-temporal data, consisting of five concepts which are connected by *has-a* relations (represented by arrows).

- A *property set* groups *properties* that have the same *domain*. The *property set* relates to the *domain* that is shared among the involved *properties*.
- The *value* represents the values for each *item* in a *phenomenon* for a particular *property*. Each *property* has one *value*.

A *phenomenon* is related to zero or more *properties*. *Properties* represent certain traits of the *phenomenon*. All *domains* and all *values* represent the same number of individuals, i.e. each *item* has an entry in both the *domain* and the *value*.

3.1.1. The phenomenon concept

Different from most existing spatio-temporal models in which data is regarded either as consisting of individual objects/agents or as a continuous field, the conceptual data model presented here starts from the idea that everything in the world can be captured by the notion of a phenomenon. This notion is somewhat more specific than for instance the feature concept in ontology, which is used in order to represent a general and homogeneous set of things (Guarino et al., 2009). The *phenomenon* concept is used to represent sets of things that share properties, but also share spatial and temporal types.

In the conceptual data model, the concept *phenomenon* is used to describe a concrete or abstract constituent of the system that is being modelled. For example, if one models an ecosystem in which `trees` and `groundwater` flows are involved, two phenomena could be used to represent the system, `trees` and `groundwater`. A phenomenon can represent a single object, for instance the groundwater over the whole study area, or multiple objects, such as individual trees contained in the phenomenon `trees`. A phenomenon often consists of multiple individuals, which are called items in the data model. This is the case for instance with the example of the phenomenon `trees`, which includes multiple individual trees as items.

3.1.2. The property and property set concepts

A *property* describes, by means of a reference to a *value* a particular property, attribute or trait of a *phenomenon* for a given spatio-temporal domain. A *property* is included in exactly one *property set* and has exactly one *value* concept (see Section 3.1.4).

A *phenomenon* can have one or more *property sets*. A *property set* can be thought of as a set of *properties* sharing a particular spatio-temporal domain, i.e. the area and time period for which the *property* exists. The *property set* refers to exactly one *domain* and a *domain* belongs to exactly one *property set*. For example, the *phenomenon* `birds` has a property set `foraging area`, which refers to the spatio-temporal domains of the `foraging areas` of each of the individual birds in the phenomenon. The property set groups several properties sharing the spatio-temporal domain. The properties `residence time and amount of food collected` for example, share the same spatio-temporal domain and hence are grouped into the property set `foraging area`. A useful consequence of the binding between properties by means of a property set is that change in the spatio-temporal traits of a property set affects all properties the property set entails.

3.1.3. The domain concept

The *properties* of *phenomena* always have a spatial and temporal location and/or extent. For instance, the maximum height of trees is defined at certain spatial and temporal locations, for example the location of the stem of each tree, while the seed dispersal area of trees is defined for the spatial and temporal extents that can possibly be reached by each tree.

The conceptual data model represents this spatial and temporal definition of a *property* as the *domain*. *Properties* sharing the same *domain* are grouped into the same *property set*. The *domain* is the spatial and temporal area or period for which a *property* is defined. In the *domain* concept, for each *item* in the *phenomenon* (see section 3.1.1), a *domain item* is defined. A *domain item* is the spatio-temporal location and extent of an individual in the *phenomenon*. The spatial *domain* can consist of *points*, *lines*, *polygons*, *polyhedrons*, but can also contain discretized data, for example cells describing the extent of a *phenomenon*. Note that in this case the cells do not contain property values but merely dichotomous values describing a *phenomenon's* presence. The temporal *domain* has equivalents for the spatial types. A *timepoint* represents a particular location in time. An *interval* is a *duration* of time with reference to a *timepoint*. As can be seen from Fig. 3, the *domain* is a necessary concept for the modelling of a *property* or *property set*, i.e. if a *property* or *property set* is defined, there must be a *domain* defined.

Using the example of trees, two *domains* could be associated with the *phenomenon* `trees`, namely one domain containing a point set of tree locations, and another domain containing a set of polygons representing the tree crowns (see Fig. 4). For each `tree` item defined in the phenomenon, a point location is defined in the `tree locations` domain. The property height for example, is associated with the point domain. The property `biomass area` is associated with a polygon domain. As can be seen from Fig. 4 there is no temporal variation in the location of the tree. The temporal domain consists of bounded periods of time in which the change in value of the property is defined.

3.1.4. The value concept

Properties of real world phenomena often have meaning because of the associated value within their specific spatio-temporal definition. Such a value can be constant over time and spatial extent, such as the species of a tree, or the weight of a stone within a given temporal domain. The value can also vary continuously in space or time, such as the height of a tree crown or the attenuation of a signal over distance.

In the conceptual data model, the *value* of a *property* is represented with reference to the *domain* of the *property set* the *property* belongs to. Since the *value* is defined for a particular *domain*, a *property* has exactly one reference to a *value* concept (Fig. 3). The *value* allows for the modelling of spatial and temporal variation within the *domain*. It does so by supporting the possibility of discretization. If there is spatio-temporal variation for an *item*, the *value* discretizes space and time, containing a value for multiple *x, y, t*. If there is no spatio-temporal variation, the *value* consists of a single value for each *item*.

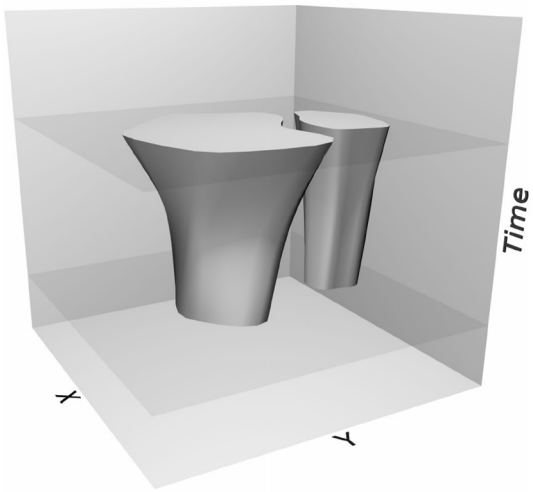
In the example of the biomass of trees, the *value* will represent the `biomass` of the `crown` of a tree individual as shown in Fig. 5.

4. Addressing the requirements

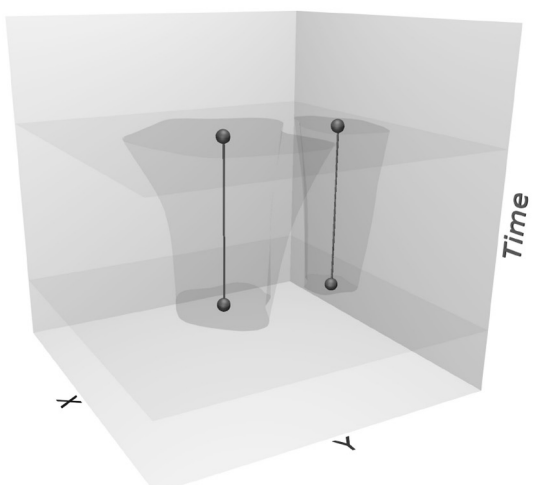
In this section we explain how the conceptual data model addresses the requirements put forward in Section 2 and provide examples supporting our rationale.

4.1. Representing fields in the conceptual data model

The conceptual data model is capable of representing classical examples of fields. For instance, a Digital Elevation Model (DEM), storing the level of the area surface as a two-dimensional discretized field of topographical height values over the whole study area, is represented by defining a *phenomenon* `area`, which has a property set `surface`, containing the property level (Fig. 6). In addition, the same property set can contain other properties that exist in the same domain, such as `soil type` used here. One

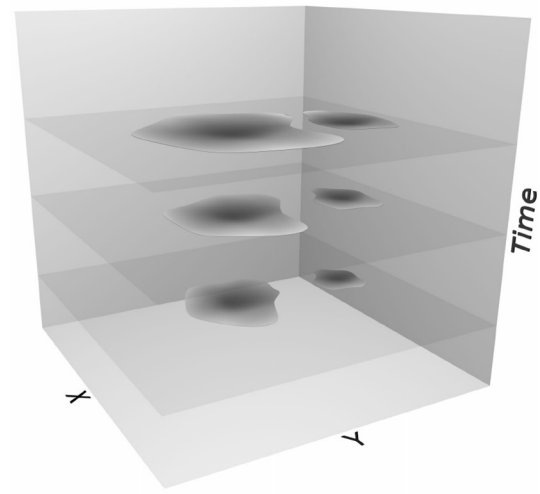


(a) Tree crown extent

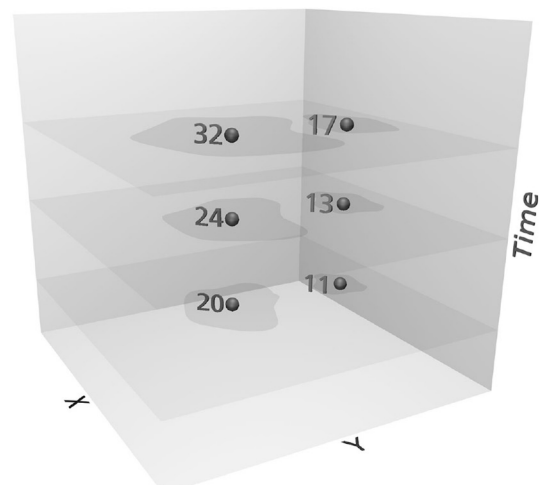


(b) Tree stem location

Fig. 4. Spatio-temporal *domain* of the `trees` phenomenon in two spatial dimensions and time. (a) `Biomass area` domain, extent of tree crowns, growing continuously in time and thus extending in the spatial domain over time. (b) `Tree locations` domain, point locations of tree stems through time. The `tree stems` are stationary in space, while the `tree` extent (shown in (a), and in (b) copied as background) increases.



(a) Tree crown biomass



(b) Tree stem height

Fig. 5. Examples of different *values*. (a) Biomass of the tree crowns represented as a two dimensional array *value*. (b) `Height` of tree stems represented as scalar value. These values belong to the `tree` items represented by the location of the `tree stems`.

domain, `extent`, is defined and related with the `surface` property set, describing the area over which soil data is available. The elevation of the `surface` is represented by means of the property `level`. The related value contains the actual DEM in the form of a two dimensional array. The `soil type` property stores for the same area a field containing the soil types present in the `area` phenomenon. It contains data for the same area as the DEM and is thus stored in the same property set `surface`. The representation works also for three dimensional cases. The property set `subsoil` is related with a `porosity` property, which defines the porosity of the subsoil in three dimensional space, `3D blocks`, to be used for instance in a groundwater flow model.

4.2. Representing agents and agents with internal variation

Agents are represented using the same layout as fields. For example, Fig. 7 illustrates how attributes of individual `trees` represented as agents can be modelled. Different from the case of the field, now the phenomenon consists of several items, all having a domain and a value defined for each property. The tree `stem height` is a property that is defined for each tree at its `location` in space and time, given by a single `scalar` value for each `tree`.

Agents with internal variation can be represented using a combination of the agent and the field representation. In this case, the *value* consists of a discretization, allowing a field-based representation for each *item* in the *phenomenon*. An example is the `crown` of a `tree` agent (Fig. 8). The biomass may vary within a certain area, i.e. the area covered by the `crown`. Therefore, the property set `crown` is connected with a domain `extent` representing the crown covered by each tree and the value has for each item a field discretized as a two- or three-dimensional array storing the values of the `biomass` of the tree inside the tree `crown`.

4.3. Representing networks between agents

To represent networks, the same concepts from Fig. 3 are used.

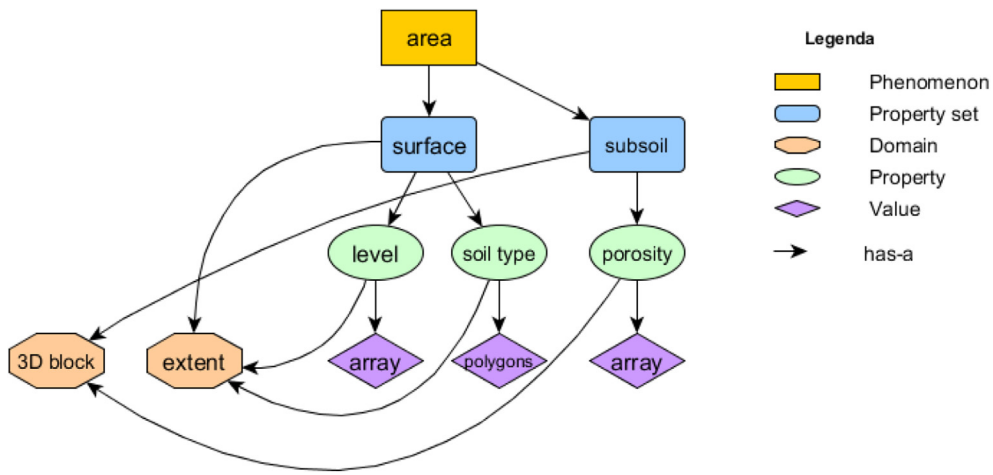


Fig. 6. A study area with exemplary properties as represented by the conceptual data model. The *property set* `surface` combines two properties sharing a spatio-temporal domain. The property `porosity` uses a different domain and is therefore put in a different property set.

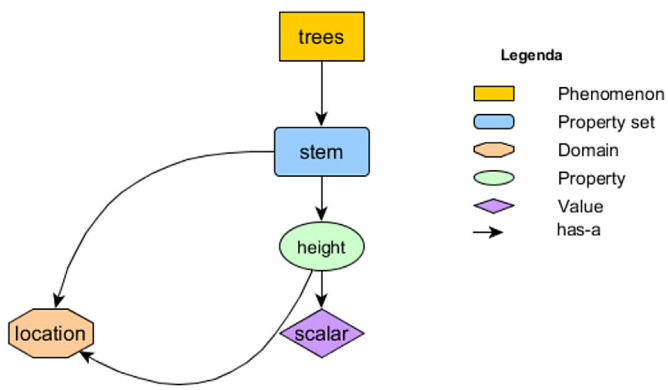


Fig. 7. *Phenomenon* `trees` consists of several individual trees. The point domain and the value representing the `stem height` define for each tree the property value and the `location` in space-time.

This is possible since networks have properties (also called weights) and domains in which they are valid. From the perspective of the modeller it is an advantage that networks can be represented by the same concepts as other phenomena because it means consistency in data storage and modelling language.

Network *phenomena* associate *items* of *phenomena* with each other. This association can occur between *items* within a *phenomenon*, or of *items* between different *phenomena*. Fig. 9 schematically illustrates a network. The yellow dots represent for instance *items* in the *phenomenon* `people`.

The blue arrows represent *items*, associating `people` to each other. In network context, these are the nodes and edges respectively. The association is made by means of the properties in the phenomenon, which can be seen as edge weights in a network. The property set in a relation phenomenon groups weights sharing the same domain of the relation. The value of a property defines for each item (that is, each edge in the network) which items from the associated phenomena (that is, the nodes) are involved and what the value (that is, the weight) of the edge is. Note that the value thus contains references to other phenomenon items.

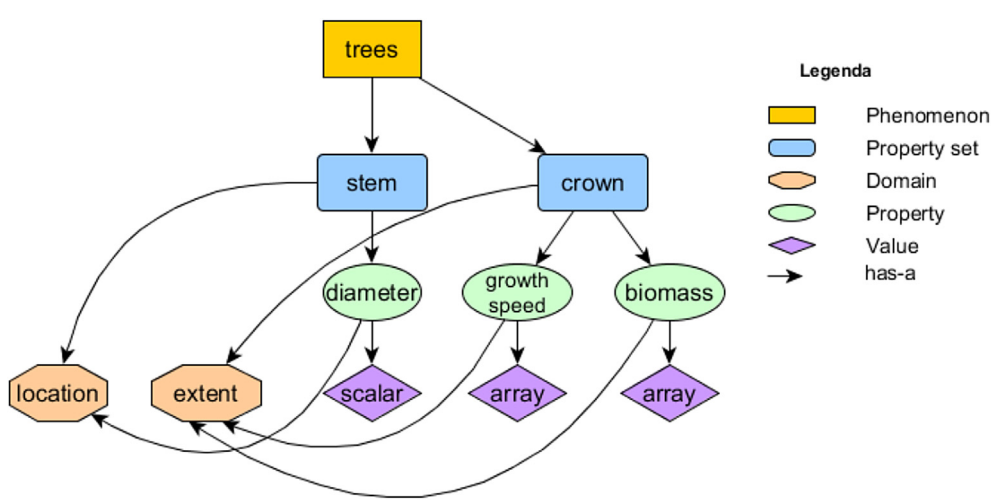


Fig. 8. Agents with internal variation, i.e. variation within their extent. The combination of a bounded area in the *domain* with a discretized field in the *value* represents for each agent the internal variation. This internal variation can exist next to other representations within the *phenomenon*, as shown by the `stem` property set, which is a single `location` for each tree.

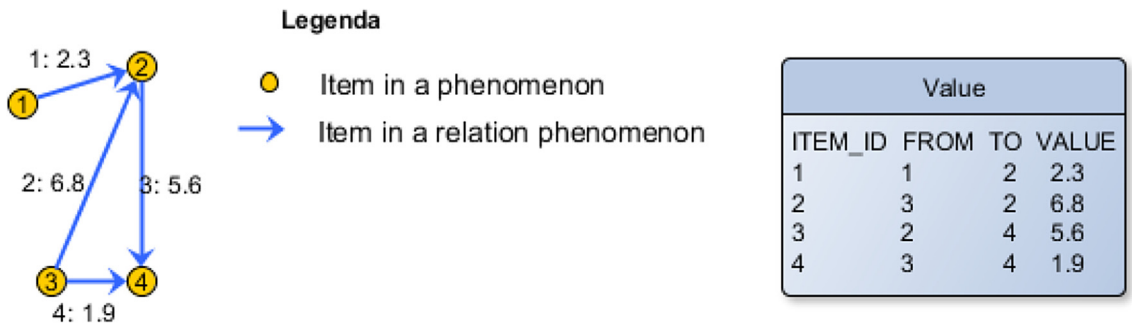


Fig. 9. Representation of a network. The arrows represent *items* in the network *phenomenon*. The dots represent *items*, for instance `persons`, in a different phenomenon. The numbers in the dots represent the item ID's. The arrows also have ID's followed by a value, for example the `amount` of information exchange. The table shows the data contained in the value.

5. Software implementation and case study

This section shows the design and implementation of a physical data model following the concepts of the conceptual data model presented in this paper. The conceptual data model has been implemented in HDF5 (The HDF Group, 2016), with API's written in C, C++ and Python. We selected HDF5 for implementation because of its flexibility in defining hierarchical models, portability, interoperability with other programming languages, and support for parallel I/O (PHDF5, 2015). The HDF5 framework is currently often used in among other things data intensive research (e.g. Hinsen, 2012; Klein and Taaheri, 2007).

5.1. Software stack

The conceptual data model presented in this paper gives a uniform representation of various kinds of data at a high level of abstraction, where the *phenomenon* concept is the highest level of abstraction. The *phenomenon* may contain a considerable range of different types of data. For instance, Fig. 8 shows the `trees` phenomenon containing both data at point locations and data related to a spatial extent, described by the `location` and `extent` domains, respectively. Examples of other data that can be stored in a phenomenon are relations and discrete time series. We call these kinds of data in the conceptual data model data types. Data types are any valid combination of domain and value, such as agent, field and relation.

While different data types are treated equally at the abstraction level of the *phenomenon*, for performance reasons they each require a specific data structure for physical implementation. Thus, an HDF5 *data model* has been designed for each data type, resulting in a set of HDF5 data models (e.g. for agents and fields) embedded in our physical data model (Fig. 10). These data models are constructed and managed by a layer on top of HDF5, the *data model interface* (Fig. 10), which is written in C and C++. This layer provides an interface between our conceptual data model, which stores data as *phenomenon*, *property set*, *property*, *domain* and *value* (see Fig. 3), and the various data models in HDF5. Together, the HDF5 *data models* and the *data model interface* form the *physical data model*.

The physical data model is about storage and retrieval of data from disk. It would be inefficient in a modelling context to work directly on the physical data model due to the file access overhead. Hence an *in-memory data model* (see Fig. 10) has been built to read and write data from and to the *physical data model* when needed. The in-memory data model reflects the conceptual data model and allocates the data retrieved from the physical data model in memory. The in-memory data model allows the quick retrieval of data, which is often necessary in calculations.

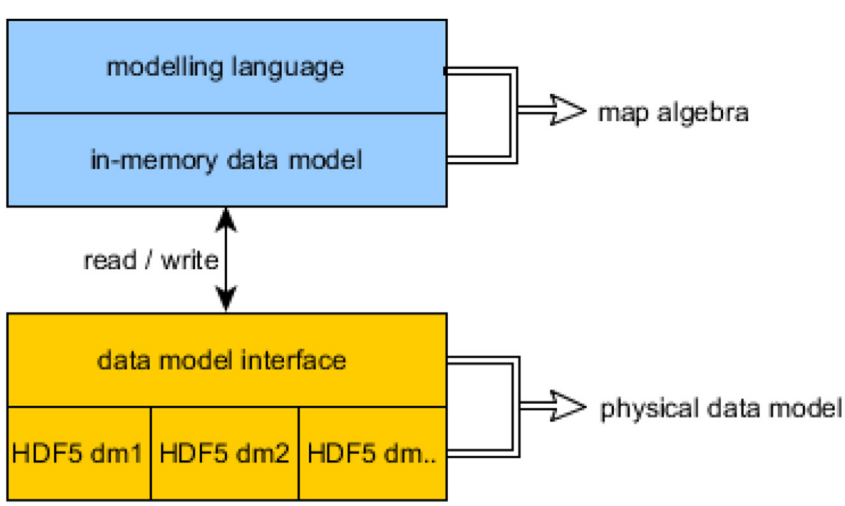


Fig. 10. Data model implementation. The implementation consists of several HDF5 models (e.g. HDF5 *dm1*) which are created and maintained by means of the *data model interface*. Together, these blocks form the *physical data model*. The *map algebra* block consists of an *in-memory data model* reflecting the conceptual data model which reads and writes data from the *physical data model*. On top of the *in-memory data model* is the *modelling language* which is the interface to the modeller.

In addition, the in-memory data model is the basis for a *modelling language* (Fig. 10), which has been discussed earlier (Section 2) as a requirement of the conceptual data model. The modelling language provides modellers with map algebra like language constructs that manipulate the data in the data model. The modelling language prototype is currently implemented in Python. The behaviour of an operation depends on the operation class and the data type of the arguments. Examples of operation classes are local operations, movement and selection (or querying). In the modelling language, implementation details are hidden from the modeller, who only needs to think at the level of the conceptual data model in order to understand how to use an operation.

5.2. Prototype implementation of key data types

We demonstrate the implementation of two data types that enable integrated field-based and agent-based modelling (Fig. 1), namely one data type for modelling a single, gridded field, and one for modelling a set of mobile agents. The field data type has one or more continuous properties, which are discretized in equally sized cells. The cell values are stored at discrete time steps. The data type consisting of a set of mobile agents stores for each agent a location in space and attributes, which are both updated at discrete time steps. Note that these data types enable only a particular type of integrated field-based and agent-based modelling, as they are restricted to, for instance, fixed time step modelling. Other data types for field-based and agent-based modelling are implemented as well, for instance those representing a set of agents having a spatial extent, or with a domain discretized by irregular time steps. The explanation of the implementation of all data types however requires the design of a formal taxonomy of data types, which is beyond the scope of this paper. Therefore, we restrict ourselves to the two data types used in a case study.

The two data types are each implemented by a specific HDF5 data model (*HDF5 dm1* and HDF5 *dm2* in Fig. 10). These HDF5 implementations make use of the HDF5 objects, which are *groups*, *attributes* and *datasets* (The HDF Group, 2016). An HDF5 group is the organizing object, and it can contain (multiple) other groups, datasets or attributes. HDF5 datasets objects contain n-dimensional arrays of data. HDF5 attributes contain data about HDF5 groups or HDF5 datasets. The structure of HDF5 data sets is analogous to a file system, in which groups are 'folders' and datasets are the 'files'. Using these HDF5 objects, we have designed the HDF5 implementation of the two data types discussed here, the field and the set of agents. We make use of HDF5 groups and HDF5 datasets in order to store the data. Appendix A discusses the implementation in detail and shows that the HDF5 file structure for both data types is very similar, except for a few details.

A software layer, the *data model interface* (Fig. 10), written in C and C++, creates the HDF5 data models, and contains operations to update and retrieve data. The software layer contains templates of the various data types to allow for the maintenance of the HDF5 data models. A less extensive version written in Python is used in this case study.

5.3. Case study model

The case study model represents grass biomass growth and grazing cows. We use a highly simplified representation of the processes retaining key elements of interaction between fields and agents, as our case study is intended only to illustrate the use of the data model and map algebra language.

The change in biomass $B(kg{\cdot}m^{-2})$ at a location is represented by the logistic growth equation with grazing and dispersion components (Karssenberg and Bierkens, 2012; Noy-Meir, 1975):

$$\frac{dB_{x,y}}{dt} = rB_{x,y}\left(1 - \frac{B_{x,y}}{k}\right) - g + dD_{x,y} \tag{1}$$

where $D_{x,y}$, the dispersion, of a cell (x,y) is determined by

$$D_{x,y} = B_{x,y-1} + B_{x,y+1} + B_{x+1,y} + B_{x-1,y} - n_{x,y}B_{x,y} \tag{2}$$

In Equation (1), k is the carrying capacity 10 ($kg{\cdot}m^{-2}$), and r is the growth rate 0.02 (day^{-1}), and g is grazing ($kg{\cdot}m^{-2}{\cdot}day^{-1}$). Biomass is represented by a discretized field, using raster cells with a constant size of $1{\cdot}m^2$. For each grid cell, Equation (1) is solved at a set of time steps with a duration of one day. Biomass removal is represented by $i = 1, 2, .., n$ mobile, foraging, cows. Each cow eats biomass at a maximum rate of $g_i = a_i{\cdot}w_i{\cdot}kg{\cdot}day^{-1}$, with a_i, a parameter 0.0008 (day^{-1}), and w_i, the weight of the cow (kg). The grazing g in Equation (1) at a particular grid cell and time step is the sum of the g_i values of all cows i in the grid cell at that time step, divided by the cell area, which is in this case 1. Grazing at a particular grid cell stops when biomass B is less than 0.01 $kg{\cdot}m^{-2}$. The dispersion $dD_{x,y}$ is calculated by summing for each cell the biomass of the directly surrounding cells and subtracting the biomass at the cell itself multiplied by the number of surrounding cells. This number is four for all cells except cells at the border. The parameter d is the dispersion rate.

At each time step the cows move. For simplicity, the movement of the cows is random, based on a bivariate (x, y) normal distribution with the current location of each cow as the mean and a σ of 30 m. However, the cows only move to a certain cell if the cell contains enough biomass, that is, if there is at least 0.5 $kg{\cdot}m^{-2}$ available. If this amount is not available, a new random location is calculated until the requirement is met.

The cows increase per time step in weight with an amount of 0.3% of their current bodyweight, until the maximum weight of 750 kg is reached.

5.4. Implementation of the case study model

The implementation of the case study model consists of two *phenomena*, namely `grass` of the field data type, and `cows` of the agent data type, shown in Fig. 11. The `grass` phenomenon contains one `biomass` property at the start of the simulation. The `cows` phenomenon contains a `weight` property. The model is programmed using the modelling language (Fig. 10).

The modeller uses a syntax that is related to the concepts in the data model. Following map algebra, the form of the operations in the language is $result = operation(arg_1, \ldots, arg_n)$, with *result* and arg_i a *phenomenon*, *value* or *domain*.

Listing 1 shows the code of how the grazing model is programmed by the modeller. Model parameters are set as Python variables in lines 1 and 2, and lines 4 to 6. Two initial data sets exist, one containing the `cows` phenomenon and one containing the `grass` phenomenon. These datasets are loaded using the function $phenomenon = read(HDF5\_dataset)$, as is done in lines 8 and 9. Appendix A provides a description of the file structure of these HDF5 datasets. The variables `grass` and `cows` now have the initial value of the data in the datasets stored in the in-memory data model, i.e. the initial state before the time updates start. Dot notation is used to access these components, in the order phenomenon.property_set.property. For example, `grass.area.biomass` gives access to the value of the property `biomass` in the property set `area` of the phenomenon `grass`. Also, because a property set

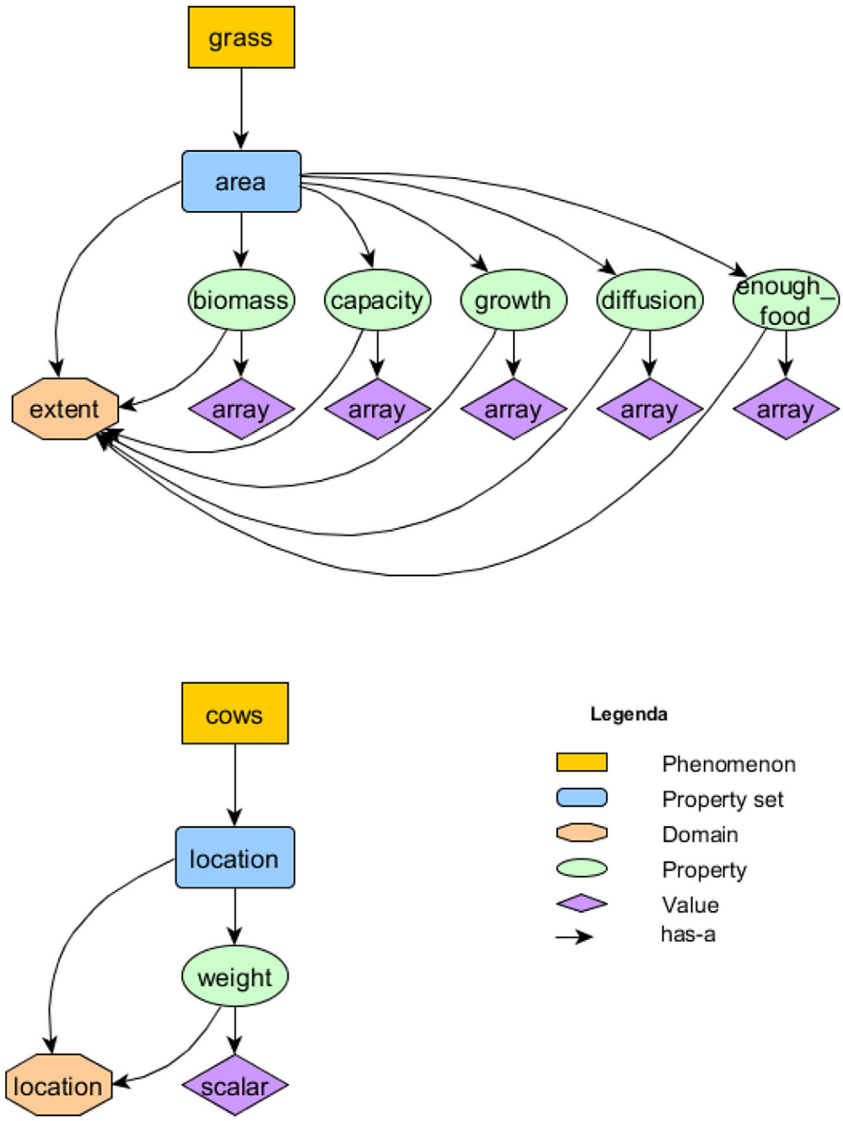


Fig. 11. Conceptual data model of the `grass` and the `cows` phenomenon for HDF5 implementation (see Appendix A) at the end of the simulation of the case study. At the beginning of the model run, in the `grass` phenomenon only the `biomass` property exists. In the `cows` phenomenon, the number of properties does not change.

refers to exactly one domain, `grass.area` provides directly access to the domain of the property set `area`.

The model runs for a specified number of time steps with a duration of one day using standard Python iteration (line 13), where the number of time steps is defined outside this script. Lines 15 to 26 represent all terms of Equation (1). Note that the operations are applied to all cells on the biomass map. For example, line 15 retrieves the biomass using `grass.area.biomass` and divides the value of all cells by the carrying capacity. The result of the operation is assigned a new property, `capacity`, in the same property set `area`.

The last term in Equation (1), diffusion of biomass, is represented by the *window* operation in line 20, which aggregates over a spatial manhattan neighbourhood (all directly neighbouring cells next to a cell). The inputs of the calculation are the current biomass (calculated on line 18), the diffusion parameter d (line 6) and the number of neighbours for each cell in the `grass.area.biomass`. This last map is calculated in line 11 with a window(property, aggregation method, neighbourhood) operation counting for each cell the number of cells at manhattan distance. This means that all cells are assigned four neighbours except cells at the border of the map, which have two or three neighbours. The result of the calculation at line 20, `grass.area.diffusion` is added to the biomass map `grass.area.biomass` (line 24).

In line 26, the `cows` phenomenon is used to calculate how much the cows eat. The cows eat an amount of biomass proportional to their weight, which is 0.008 percent. This amount, divided by the cell area (which is in this case 1 m^2) is subtracted from the biomass of the cell where each cow resides. Because operations apply to all items in the phenomenon, the operation *cows.location.weight*$*$0.0008 is applied to all cows. Line 26 thus combines the agent and field data types by calculating the new value of the field on the basis of the value of the agent data type. The maximum(scalar, property) operation takes for each cell in biomass the maximum of the cell value and 0.01. This represents the model assumption that grazing does not occur below a biomass of 0.01 $kg{\cdot}m^{-2}$.

Line 31 increases the bodyweight of each cow by multiplying the weight *property* of each cow with 1.003. To avoid that the cows' weight continues to increase, the operation *minimum(scalar, property)* assigns the new weight or 750 to the

```
1  grass_datapath = '../data/grass.hdf5'
2  cows_datapath = '../data/cows.hdf5'
3
4  growth_rate = 0.02
5  carrying_capacity = 10.0
6  d = 0.01
7
8  grass = read(grass_datapath)
9  cows = read(cows_datapath)
10
11 grass.area.nr_neighbours = window(grass.area.biomass, 'count', 'manhattan')
12
13 for t in range(1, nr_of_timesteps):
14
15     grass.area.capacity = 1 - grass.area.biomass / carrying_capacity
16     grass.area.growth = grass.area.capacity * growth_rate *
17             grass.area.biomass
18     grass.area.biomass = grass.area.biomass + grass.area.growth
19
20     grass.area.diffusion = d * (window(grass.area.biomass, 'sum', 'manhattan') -
21                                 grass.area.nr_neighbours *
22                                 grass.area.biomass)
23
24     grass.area.biomass = grass.area.biomass + grass.area.diffusion
25
26     grass.area.biomass = maximum(0.01, grass.area.biomass -
27                                 (cows.location.weight * 0.0008))
28
29     grass.area.enough_food = grass.area.biomass > 0.5
30
31     cows.location.weight = minimum(750, cows.location.weight * 1.003)
32
33     cows.location = gaussian_move(cows.location, grass.area.enough_food, 30)
34
35     write(grass, grass_datapath, t)
36     write(cows, cows_datapath, t)
```

Listing 1. Grazing model programmed in Python using the map algebra modelling language.

`cows.location.weight` property, for each `cow`.

The movement of the cows is represented by the move operation *gaussianmove(agent_property_set, field_property_set, sigma)* (line 33). This operation calculates the new location of each cow by taking its current x and y coordinate and uses these coordinates as the mean in a bivariate (x, y) normal distribution. The standard deviation is provided by the modeller. The move operation generates a new random x and y coordinate on the basis of this distribution. An underlying boolean field, `grass.area.enough_food`, calculated in line 29, determines whether this new location is possible, namely if the biomass exceeds 0.5 $kg \cdot m^{-2}$. If not, the gaussianmove operation calculates a new location until the location is valid.

The new values that are calculated during the model run are stored in the in-memory data model. We often want to store the (intermediate) model data to disk. Thus, lines 35 and 36 write the content of the in-memory data model as a new time step to the HDF5 files stored at disk. The new data values become the current state in the in-memory data model at the following time step iteration.

Fig. 12 shows the state of the model stored to disk at several time steps. Initial biomass is assumed close to zero. It can be seen that the `biomass` increases while the `cows` wander randomly over the area. In the beginning, `cows` are concentrated in a single area. As the simulation continuous, the `cows` search for patches (cells) with enough food. While the simulation proceeds and the biomass increases, the cows seem to form new smaller clusters.

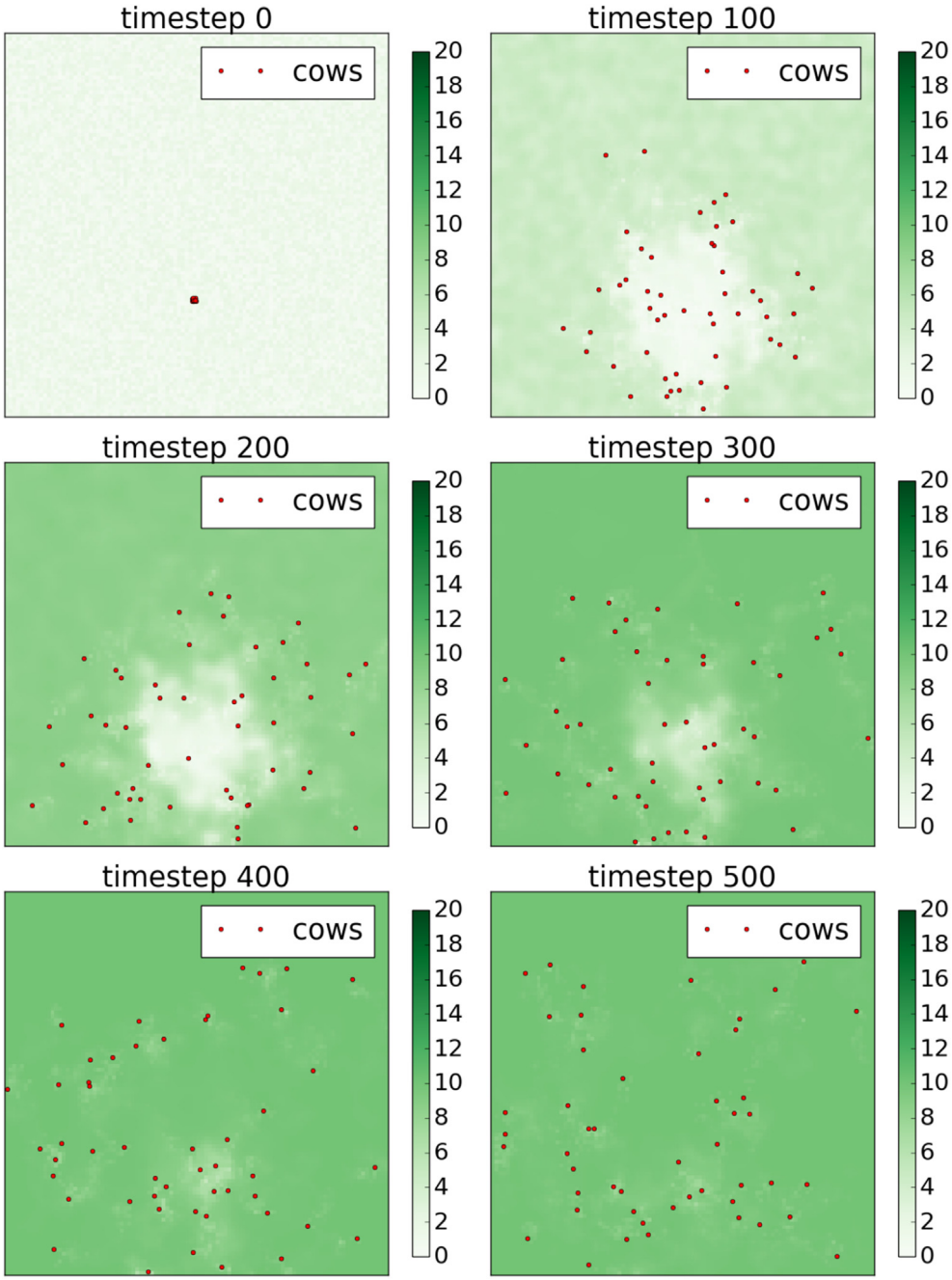


Fig. 12. grass.area.biomass (shown as a green field) and cows.location (shown as red dots) at 0, 100, 200, 300, 400 and 500 days. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6. Discussion and conclusion

We introduced a new data model for representing several types of data commonly used in environmental modelling. We have shown that this data model is able to combine field and agent data and is suitable for a modelling language that is built upon the concepts of map algebra.

A number of model building frameworks already exist and interfacing between frameworks is in principle possible. However, our conceptual data model can be a valuable contribution to the modelling community. For instance, a number of authors (Athanasiadis, 2013; Kosar et al., 2010; Degenne et al., 2009) have stressed that it is advantageous to programmers and modellers if the semantics of the language they use is near to their application domain. This is exactly what is provided by our data model: the modeller gets access to fields, agents and relations, and is capable of manipulating this information through a single conceptual representation of the data. Also, it is stressed in the literature that domain-specific languages are important for more efficient and effective model implementation as they take away programming complexity from the user (Athanasiadis, 2013; Mernik et al., 2005). This is particularly relevant for modellers of spatio-temporal systems because the systems being modelled are becoming increasingly complex and thus require more complicated calculations and

coding. Our data model is, as shown here, a relevant step in this direction, as it can be used as argument in a map algebra like domain-specific language. Map algebra is known to be a concept that can easily be grabbed by domain specialists without a strong programming background (Mennis, 2010).

The value of the data model is thus to offer an approach for modelling fields and agents. This is not to say that the data model prescribes the way a modeller represents a certain problem. What representation to choose given the research question and the available data is still a task of the modeller. Our conceptual data model does not prescribe how to represent data in a particular modelling question. For example, in the case of modelling land-use, it is the modeller who decides to represent land-use either as a field, where a nominal attribute value represents the land use type, or alternatively as set of agents, where each individual field is an agent. In a similar fashion, it is up to the modeller to choose the representation of biomass, as a continuous field of biomass values for each pixel, for instance derived from a remote sensing image, or as a set of agents, where each tree is represented by an agent. Our data model can accommodate these different representations, which can coexist, and which can be chosen according to the preferences of the modeller. This flexibility is advantageous because the choice for a representation is usually specific for a particular study, and dependent on, for instance, the research objectives and the data availability.

The idea of integrating the modelling approaches is not new, and we have build upon existing work on conceptual data models for geographic data representation (see Section 2). For instance, as is the case for the OGC specification (OGC, 1999), we represent relations separate from the nodes in the relation. Also, like Voudouris (2010) we allow the representation of phenomena having properties that are both field-like and object-like, and like Grignard et al. (2014) we aim at providing an easy to use modelling language. Still, our conceptual data model is distinguished from existing work in several ways. For example, having multiple individuals with different domains included in a single *phenomenon* is new. Implementations of agents having internal variation do currently not exist in the agent-based frameworks without the need of ad hoc workarounds (e.g. NetLogo (2012), GAMA (Grignard et al., 2014), Mason (Sullivan et al., 2010), Repast (North et al., 2013) and TerraME (Carneiro et al., 2013)). Also, from the point of view of language design, our work is distinguished from other examples in the literature. For instance, it differs from recent work by Kuhn (2014) which aims at developing generic queries for geographic information, but still uses two different representations for objects and fields.

The prototype of the data model and modelling language was demonstrated in Section 5. The implementation shows that the conceptual data model can be implemented as a physical data model with a modelling language on top to manipulate the data stored in the data model. The prototype reveals advantages of the work presented in this paper. Firstly, it shows that it is possible to integrate different data types at a high level of abstraction by implementing several representations in the lower level physical data model. Secondly, using the modelling language it is possible to manipulate data in the physical data model. An example of this is that operations implicitly iterate over all items in the *phenomenon*, as is exemplified by the addition of two maps and the *window* operation (see line numbers 18 and 20 in Listing 1). Thirdly, the language is capable of performing operations on combined data types, e.g. fields and agents, for example subtracting biomass from the field on the basis of the weight of the cows, and the *move* operation (see line numbers 26 and 33 in Listing 1).

Current and future research needs to focus on a number of conceptual and implementation related issues. Ongoing research includes the definition of physical data models in HDF5 for the many other data types that can be derived from the conceptual data model. Two data types (field and agent) where discussed in Section 5, which are being extended to more advanced data types such as agents having internal spatial variation. Current research also focuses on supporting high-performance computing of any dynamic environmental model developed with the data model and language introduced here by using the parallel computing version of HDF5 (PHDF5, 2015). In addition to this current work, a number of important challenges remain for future research. One is that the incorporation of time in the conceptual data model poses questions regarding temporal modelling. Modelling and simulation of continuous time and uncertainty should also be included in our framework. Another open issue regarding the modelling language is the extension of the set of intuitive generic operations that are meaningful given particular combinations of data types. The modelling language (Section 5) now consists of a limited number of operations and interactions. As new data types are being developed, the number of combinations of arguments in operations increases, meaning that an approach to assess valid combinations of arguments is needed.

7. Software availability

Scripts with the data model implementation, modelling language operations and case study model are available.

Contact: m.p.debakker@uu.nl
Instructions and download at: http://pcraster.geo.uu.nl/projects/developments/generic-modelling-of-fields-and-agents/
Direct download at:
https://github.com/pcraster/datamodel_prototype/releases

Required software

Operating system: Linux, Windows
Python 2.7 (http://www.python.org/) with the Numpy, Matplotlib and h5py modules.

Acknowledgements

We thank Dr. Judith Verstegen (University of Muenster) for her helpful comments. We also thank two anonymous reviewers from iEMSs2014, and three reviewers from Environmental Modelling & Software for their comments on previous versions of this paper. The work presented here is partly funded by EIT Climate-KIC, the Netherlands and the Global and Geo Health Data Centre (Utrecht University, the Netherlands).

Appendix A. HDF5 structure for field and agent data types

In this appendix, the structure of the HDF5 files is explained.

A.1. General structure of the data model in HDF5 files.

We have designed the HDF5 implementation of the field and the set of agents data types. We make use of HDF5 groups and HDF5 datasets in order to store the data. HDF5 groups represent *phenomenon, property set, property* and *domain*. This implies that for each *property set* or *property* a new subfolder exists, which results in a hierarchical folder structure. This hierarchical structure is

illustrated in Figs. A.13 and A.14. HDF5 datasets represent the *value* concept from the conceptual data model. Furthermore, HDF5 datasets are used to store the data contained in the *domain* and discretization information. Note that if, for example, a new *property* is created during the model run, a new HDF5 group is added to the structure.

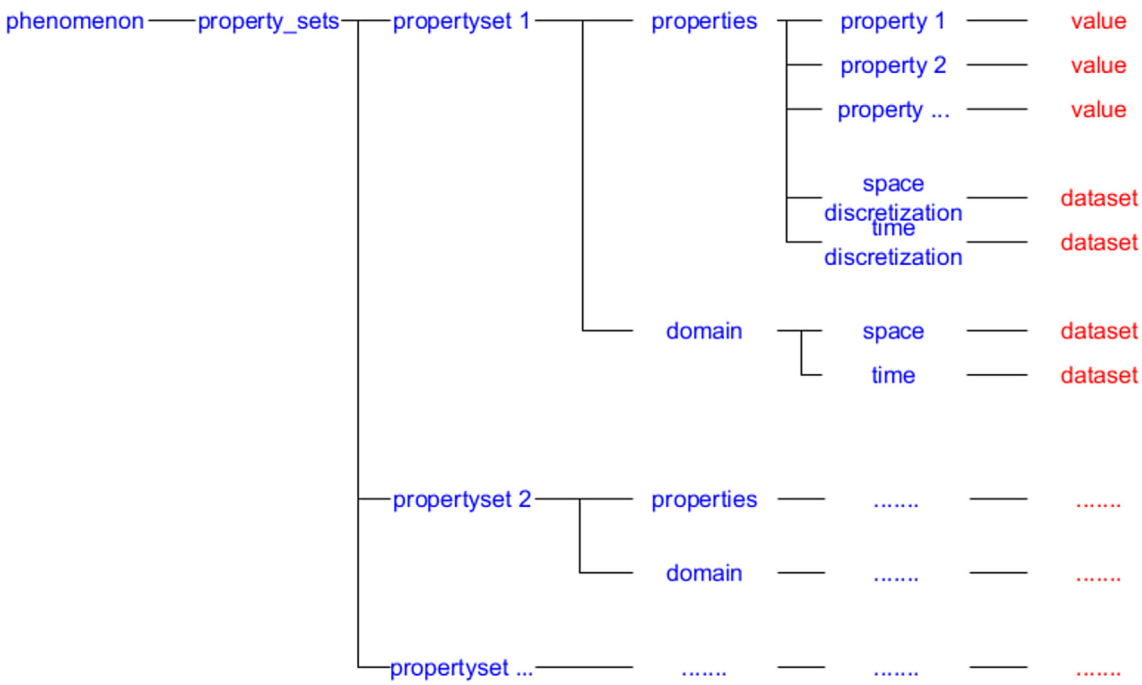


Fig. A.13. Hierarchical structure of HDF5 file of a field data type. Text in blue represents HDF5 groups and text in red represents HDF5 datasets. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

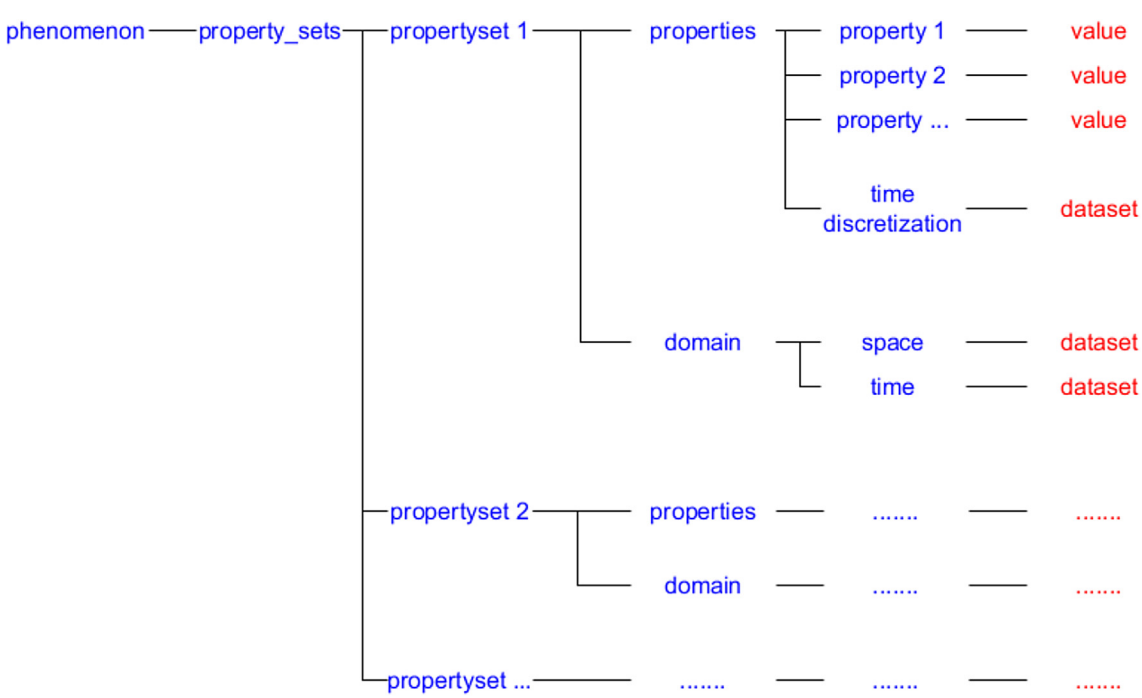


Fig. A.14. Hierarchical structure of HDF5 file of a agent data type. Text in blue represents HDF5 groups and text in red represents HDF5 datasets. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In Figs. A.13 and A.14 the concepts of the conceptual data model are recognizable. Some extra datasets are necessary to store additional information on spatial and temporal dimensions of the data (*space* and *time* HDF5 groups and related datasets in both figures). Furthermore, for the field data type, we must store how the field is discretized. This is done by means of the *space discretization* HDF5 group and related dataset. The temporal discretization HDF5 group and related dataset is present for both the agent and the field data type. Because differences between the data types is only made in the *domain* and *value*, the general HDF5 structure is almost equal for both the field and the set of agents. The layout of the HDF5 dataset, which is a *n*-dimensional array, however differs somewhat. Since the field data type only consists of one *item*, this *item* is represented by one HDF5 dataset, a 2-dimensional array. In the case of the set of agents data type, the HDF5 dataset represents multiple *items*, that is one, the first dimension of the array represents the *item*, while the second dimension represent the *value*.

A.2. HDF5 structure for `grass` *and* `cows` *phenomena from the case study.*

The figures show the structure of the files after running the model, The figures show the structure of the files after running the model. Fig. A.15 thus contains all properties that were created while executing the model.

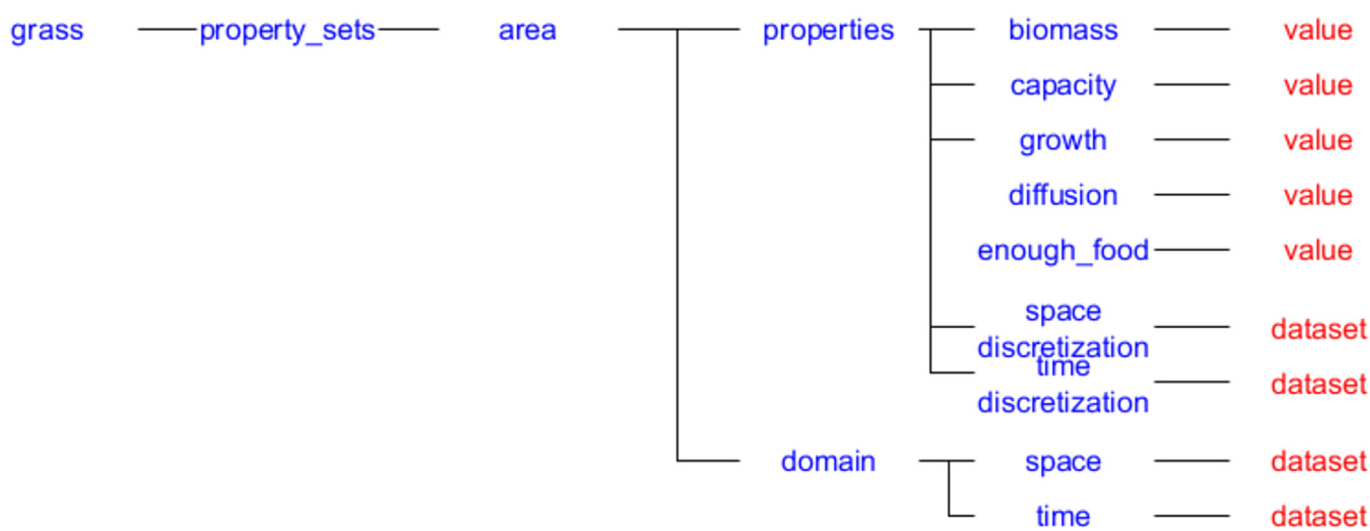


Fig. A.15. The structure of the HDF5 file grass.hdf5 after running the model (Listing 1 in the main text).

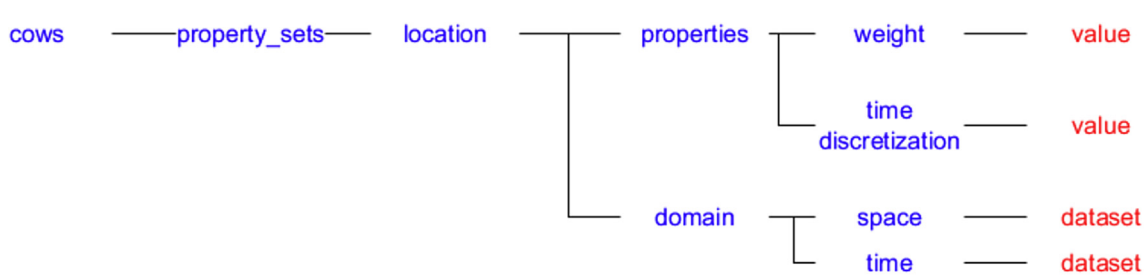


Fig. A.16. The structure of the HDF5 file cows.hdf5 after running the model (Listing 1 in the main text).

References

Abdou, M., Hamill, L., Gilbert, N., 2012. Designing and building an agent-based model. In: Agent-based Models of Geographical Systems. Springer, pp. 141–165.

Altreva, 2016. Adaptive Modeler. http://www.altreva.com/.

Athanasiadis, I.N., 2013. A Roadmap to Domain Specific Programming Languages for Environmental Modeling Key Requirements and Concepts, pp. 27–32. Proceedings of the 2013 ACM workshop on Domain-specific modeling.

Baumann, P., 1994. Management of multidimensional discrete data. VLDB J. 3 (4), 401–444.

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., Widmann, N., 1999. Spatio-temporal retrieval with RasDaMan. In: Proceedings of the 25th VLDB Conference. Scotland, Edinburgh, pp. 746–749.

Bennett, D. a., Tang, W., 2006. Modelling adaptive, spatially aware, and mobile agents: elk migration in Yellowstone. Int. J. Geogr. Inf. Sci. 20 (9), 1039–1066.

Berger, T., 2001. Agent-based spatial models applied to agriculture : a simulation tool for technology diffusion, resource use changes and policy analysis. Agric. Econ. 25, 245–260.

Bithell, M., Brasington, J., Richards, K., 2008. Discrete-element, individual-based and agent-based models: tools for interdisciplinary enquiry in geography? Geoforum 39 (2), 625–642.

Bonabeau, E., 2002. Agent-based modeling: methods and techniques for simulating human systems. Proc. Natl. Acad. Sci. 99 (Suppl. 3), 7280–7287.

Brown, D.G., Riolo, R., Robinson, D.T., North, M., Rand, W., 2005. Spatial process and data models: toward integration of agent-based models and GIS. J. Geogr. Syst. 7 (1), 25–47.

Brown, P.G., 2010. Overview of SciDB: large scale array storage, processing and analysis. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 963–968. Indianapolis, Indiana, USA.

Burrough, P.A., McDonnell, R.A., 1998. Principles of Geographical Information Systems. Oxford University Press, New York.

Câmara, G., Monteiro, A.M.V.M., Paiva, J.A., Gomes, J., Velho, L., 2000. Towards a unified framework for geographical data models. In: Geoinfo 2000. Workshop Brasileiro de Geoinformatica, Sao Paulo.

Câmara, G., Palomo, D., Cartaxo, R., Souza, M.D., 2005. Towards a generalized map algebra : principles and data types. In: Workshop Brasileiro de Geoinformatica, vol. 7. Campos de Jordao, pp. 66–81.

Carneiro, T.G.D.S., Andrade, P.R.D., Câmara, G., Monteiro, A.M.V., Pereira, R.R., 2013. An extensible toolbox for modeling nature-society interactions. Environ. Model. Softw. 46, 104–117.

Castilla-Rho, J.C., Mariethoz, G., Rojas, R., Andersen, M.S., Kelly, B.F.J., 2015. An agent-based platform for simulating complex human-aquifer interactions in managed groundwater systems. Environ. Model. Softw. 73, 305–323.

Clarke, K.C., 2007. Mapping and modelling land use change : an application of the SLEUTH model. In: Landscape Analysis and Visualisation. Springer, Berlin Heidelberg, pp. 353–366.

Cova, T.J., Goodchild, M.F., 2002. Extending geographical representation to include fields of spatial objects. Int. J. Geogr. Inf. Sci. 16 (6), 509–532.

Crooks, A.T., Castle, C.J., 2006. Principles and concepts of agent-based modelling for developing geospatial simulations. Tech. Rep. (110) UCL Centre for Advanced Spatial Analysis, London.

Crooks, A.T., Castle, C.J.E., 2012. The integration of agent-based modelling and geographical information for geospatial simulation. In: Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (Eds.), Agent-based Models of Geographical Systems. Springer, Netherlands, pp. 219–251.

Degenne, P., Lo Seen, D., Parigot, D., Forax, R., Tran, A., Ait Lahcen, A., Curé, O., Jeansoulin, R., 2009. Design of a domain specific language for modelling processes in landscapes. Ecol. Model. 220 (24), 3527–3535.

ESRI, 2015. ArcGIS Desktop.

Ferreira, K.R., Camara, G., Monteiro, A.M.V., 2014. An algebra for spatiotemporal data: from observations to events. Trans. GIS 18 (2), 253–269.

Filatova, T., Verburg, P.H., Parker, D.C., Stannard, C.A., 2013. Spatial agent-based models for socio-ecological systems: challenges and prospects. Environ. Model. Softw. 45, 1–7.

Fisher, P.F., 1999. Models of uncertainty in spatial data. In: Longley, P., Goodchild, M., Maguire, D., Rhind, D. (Eds.), Geographical Information Systems. John Wiley & Sons, inc., New York, pp. 191–205.

Fowler, M., 2010. Domain Specific Languages. Addison-Wesley Professional.

Frank, A.U., 2005. Map algebra functors temporal data 1980 (Esri 1993), 14.

Galton, A., 2001. A formal theory of objects and fields. In: Spatial Information Theory. Springer, Berlin Heidelberg, pp. 458–473.

Galton, A., 2004. Fields and objects in space, time, and space-time. Spatial cognition Comput. 4 (1), 39–68.

Galton, A., Worboys, M., 2005. Processes and events in dynamic geo-networks. In: Rodriguez, M., Cruz, I., Levashkin, S., Egenhofer, M. (Eds.), GeoSpatial Semantics (GeoS 2005). Springer Lecture Notes in Computer Science, Berlin, pp. 45–59.

Gebbert, S., Pebesma, E., 2014. A temporal GIS for field based environmental modeling. Environ. Model. Softw. 53, 1–12.

Goodchild, M.F., 2013. Prospects for a space-time GIS. Ann. Assoc. Am. Geogr. 103 (5), 1072–1077.

Goodchild, M.F., Yuan, M., Cova, T.J., 2007. Towards a general theory of geographic representation in GIS. Int. J. Geogr. Inf. Sci. 21 (3), 239–260.

Goodrich, M.T., Tamassia, R., 2002. Algorithm Design: Foundations, Analysis, and Internet Examples. John Wiley & Sons, inc.

Grignard, A., Taillandier, P., Gaudou, B., Vo, D.A., Huynh, Q., Drogoul, A., Grignard, A., Taillandier, P., Gaudou, B., Vo, D.A., Huynh, N.Q., 2014. GAMA 1. 6 : advancing the art of complex agent-based modeling and simulation. PRIMA 2013 Princ. Pract. Multi-Agent Syst. 117–131.

Grimm, V., Railsback, S.F., 2013. Individual-based Modeling and Ecology. Princeton University Press.

Guarino, N., Oberle, D., Staat, S., 2009. What is an Ontology. In: Staab, S., Studer, R. (Eds.), Handbook on Ontologies. Springer-Verlag, Berlin Heidelberg, pp. 1–17.

Hagberg, A.A., Schult, D.A., Swart, P.J., 2008. Exploring network structure, dynamics, and function using {NetworkX}. Proceedings of the 7th Python in Science Conference (SciPy2008), pp. 11–15. Pasadena, CA USA.

Hinsen, K., 2012. Caring for your data. Comput. Sci. Eng. 70–74.

Hoek, G., Beelen, R., de Hoogh, K., Vienneau, D., Gulliver, J., Fischer, P., Briggs, D., 2008. A review of land-use regression models to assess spatial variation of outdoor air pollution. Atmos. Environ. 42 (33), 7561–7578.

Holst, N., Belete, G.F., 2015. Domain-specific languages for ecological modelling. Ecol. Inf. 27, 26–38.

Jjumba, A., Dragicevic, S., 2015. Integrating GIS-based geo-atom theory and voxel automata to simulate the dispersal of airborne pollutants. Trans. GIS 19 (4), 582–603.

Karssenberg, D., 2002. The value of environmental modelling languages for building distributed hydrological models. Hydrol. Process. 16 (14), 2751–2766.

Karssenberg, D., Bierkens, M.F.P., 2012. Early-warning signals (potentially) reduce uncertainty in forecasted timing of critical shifts. Ecosphere 3 (2), 15.

Karssenberg, D., Bridge, J.S., 2008. A three-dimensional numerical model of sediment transport, erosion and deposition within a network of channel belts, floodplain and hill slope: extrinsic and intrinsic controls on floodplain dynamics and alluvial architecture. Sedimentology 55, 1717–1745.

Karssenberg, D., Schmitz, O., Salamon, P., de Jong, K., Bierkens, M.F., 2010. A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. Environ. Model. Softw. 25 (4), 489–502.

Kjenstad, K., 2006. On the integration of object-based models and field-based models in GIS. Int. J. Geogr. Inf. Sci. 20 (5), 491–509.

Klein, L., Taaheri, A., 2007. HDF-EOS5 data model, file format and library. Change 501 (July 2006), 1–56.

Kocifaj, M., Kómar, L., 2016. Modeling diffuse irradiance under arbitrary and homogeneous skies: comparison and validation. Appl. Energy 166, 117–127.

Kosar, T., Oliveira, N., Mernik, M., Pereira, M.J.V., Črepinšek, M., da Cruz, D., Henriques, P.R., 2010. Comparing general-purpose and domain-specific languages: an empirical study. Comput. Sci. Inf. Syst. 7 (2), 247–264.

Kuhn, W., 2012. Core concepts of spatial information for transdisciplinary research. Int. J. Geogr. Inf. Sci. 26 (12), 2267–2276.

Kuhn, W., 2014. What is field and object information? Vienna, Austria In: Stewart, K., Pebesma, E., Navratil, G., Fogliaroni, P., Duckham, M. (Eds.), Extended Abstract Proceedings of the GIScience 2014. GeoInfo Series Vienna 2014, pp. 96–98.

Li, X., Yang, J., Guan, X., Wu, H., 2014. An event-driven spatiotemporal data model (E-ST) supporting dynamic expression and simulation of geographic processes. Trans. GIS 18, 76–96.

Liu, Y., Goodchild, M.F., Guo, Q., Tian, Y., Wu, L., 2008. Towards a General Field model and its order in GIS. Int. J. Geogr. Inf. Sci. 22 (6), 623–643.

Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W., 2011. In: Geographic Information Systems & Science, third ed. John Wiley & Sons Ltd.

MATLAB, 2012. MATLAB. www.mathworks.com.

Mennis, J., 2010. Multidimensional map algebra: design and implementation of a spatio-temporal GIS processing language. Trans. GIS 14 (1), 1–21.

Mennis, J., Viger, R., Tomlin, C.D., 2005. Cubic map algebra functions for spatio-temporal analysis. Cartogr. Geogr. Inf. Sci. 32 (1), 17–32.

Mernik, M., Heering, J., Sloane, A.M., 2005. When and how to develop domain-specific languages. ACM Comput. Surv. 37 (4), 316–344.

Mernild, S.H., Knudsen, N.T., Yde, J.C., Malmros, J.K., 2015. Detailed spatiotemporal albedo observations at Greenland ' s Mittivakkat Gletscher. EGU General Assem. Conf. Abstr. 17, 2643.

Molteni, F., Buizza, R., Palmer, T.N., Petroliagis, T., 1996. The ECMWF ensemble prediction system: methodology and validation. Q. J. R. Meteorological Soc. 122, 73–119.

Moreno, N., Ménard, A., Marceau, D.J., 2008. VecGCA: a vector-based geographic cellular automata model allowing geometric transformations of objects. Environ. Plan. B Plan. Des. 35 (4), 647–665.

NetLogo, 2012. NetLogo multi-agent modelling environment. http://ccl.northwestern.edu/netlogo/.

North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., Sydelko, P., 2013. Complex adaptive systems modeling with Repast Simphony. Complex Adapt. Syst. Model. 1 (1), 3.

Noy-Meir, I., 1975. Stability of grazing systems: an application of predator-prey graphs. J. Ecol. 63 (2), 459–481.

OGC, 1999. The OpenGIS â„¢ abstract specification topic 8 : relationships between features. Tech. Rep. 99. http://www.opengeospatial.org/standards/as.

OGC, 2006. The OpenGIS abstract specification topic 6: schema for coverage geometry and functions. Tech. Rep. Open Geospatial Consortium Inc http://www.opengeospatial.org/standards/as.

OGC, 2009. The OpenGIS abstract specification topic 5 : features. Tech. Rep. Open Geospatial Consortium Inc http://www.opengeospatial.org/standards/as.

OGC, 2011. Ogc reference model. Tech. Rep. Open Geospatial Consortium Inc http://www.opengeospatial.org/standards/.

Parker, D.C., 2005. Information systems and use : prospects and challenges. Gis. Spatial Analysis Model. 320–341.
Parker, D.C., Filatova, T., 2008. A conceptual design for a bilateral agent-based land market with heterogeneous economic agents. Computers. Environ. Urban Syst. 32 (6), 454–463.
Parker, D.C., Manson, S.M., Janssen, M. a., Hoffmann, M.J., Deadman, P., 2003. Multi-agent systems for the simulation of land-use and land-cover change: a review. Ann. Assoc. Am. Geogr. 93 (2), 314–337.
Peixoto, T.P., 2014. The graph-tool python library. figshare. http://figshare.com/articles/graph%7b%5c%5f%7d tool/1164194.
Peuquet, D.J., 2001. Making space for time : issues in space-time data representation. GeoInformatica 5 (1), 11–32.
PHDF5, 2015. Cooperative, open, development.
Pullar, D., 2001. MapScript : a map algebra programming language incorporating neighborhood analysis. GeoInformatica 5 (1), 145–163.
Pultar, E., Cova, T.J., Yuan, M., Goodchild, M.F., 2010. EDGIS: a dynamic GIS based on space time points. Int. J. Geogr. Inf. Sci. 24 (3), 329–346.
R Core Team, 2013. R: a Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.r-project.org/.
Sbihi, H., Allen, R.W., Becker, A., Brook, J.R., Mandhane, P., Scott, J. a., Sears, M.R., Subbarao, P., Takaro, T.K., Turvey, S.E., Brauer, M., 2015. Perinatal exposure to traffic-related air pollution and atopy at 1 year of age in a multi-center Canadian birth cohort study. Environ. Health Perspect. 123 (9), 902–908.
Schelhaas, M.J., Kramer, K., Peltola, H., van der Werf, D.C., Wijdeven, S.M.J., 2007. Introducing tree interactions in wind damage simulation. Ecol. Model. 207, 197–209.
Schippers, P., Teeffelen van, A., Verboom, J., Vos, C., Kramer, K., WallisDeVries, M., 2014. The impact of large herbivores on woodland-grassland dynamics in fragmented landscapes: the role of spatial configuration and disturbance. Ecol. Complex. 17, 20–31.
Sullivan, K., Coletti, M., Luke, S., 2010. GeoMason: GeoSpatial support for MASON. Tech. Rep. (Department of Computer Science, George Mason University).
Sutanudjaja, E.H., Van Beek, L.P.H., De Jong, S.M., Van Geer, F.C., Bierkens, M.F.P., 2011. Large-scale groundwater modeling using global datasets: a test case for the Rhine-Meuse basin. Hydrology Earth Syst. Sci. 15, 2913–2935.
The HDF Group, 2016. Hierarchical Data Format, Version 5. http://www.hdfgroup.org/HDF5/.
Tomlin, C., 1983. A map algebra. In: Proceedings of Harvard Computer Conference. Cambridge, MA.
Tomlin, D., 1990. Geographic Information Systems and Cartographic Modelling. Prentice-Hall, Englewood Cliffs, NJ.
Valentine, H.T., Mäkelä, A., 2005. Bridging process-based and empirical approaches to modeling tree growth. Tree Physiol. 25 (7), 769–779.
van Deursen, A., Klint, P., Visser, J., 2000. Domain-specific languages : an annotated bibliography. ACM Sigplan Not. 35 (6), 26–36.
van Deursen, W., 1995. Geographical Information Systems and Dynamic Model; Development and Application of a Prototype Spatial Modelling Language (Ph.D. thesis). Utrecht University, The Netherlands.
van Engelen, R. a, 2001. ATMOL: a domain-specific language for atmospheric modeling. J. Comput. Inf. Technol. 9 (4), 289–303.
Voudouris, V., 2010. Towards a unifying formalisation of geographic representation: the object field model with uncertainty and semantics. Int. J. Geogr. Inf. Sci. 24 (12), 1811–1828.
Wang, X., Pullar, D., 2005. Describing dynamic modeling for landscapes with vector map algebra in GIS. Comput. Geosciences 31 (8), 956–967.
Winter, S., 1998. Bridging vector and raster representation in GIS. Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems. ACM New York, NY, USA, pp. 57–62.
Winter, S., Nittel, S., 2003. Formal information modelling for standardisation in the spatial domain. Int. J. Geogr. Inf. Sci. 17 (8), 721–741.
Yuan, M., 2001. Representing complex geographic phenomena in GIS. Cartogr. Geogr. Inf. Sci. 28 (2), 83–96.
Ziervogel, G., Bithell, M., Washington, R., Downing, T., 2005. Agent-based social simulation: a method for assessing the impact of seasonal climate forecast applications among smallholder farmers. Agric. Syst. 83 (1), 1–26.