A framework to resolve spatio-temporal misalignment in component-based modelling

Oliver Schmitz, Elga Salvadore, Lien Poelmans, Johannes van der Kwast and Derek Karssenberg

ABSTRACT

Process-based spatio-temporal component models simulate real world processes, using encapsulated process representations that operate at individual spatial and temporal discretisations. These component models act as building blocks in the construction of multi-disciplinary, multi-scale integrated models. Coupling these independent component models, however, involves aggregation or disaggregation of the exchanged variables at model runtime, since each of the component models exposes potentially different spatial and temporal discretisations. Although conceptual methodologies for spatial and temporal scaling are available, dedicated tools that assist modellers to implement dynamic spatial and temporal scaling operations are rare. We present the accumulator, a programmable general-purpose model building block executing custom scaling operations at model runtime. We therefore characterise runtime information of input and output variables required for the implementation of scaling operations between component models with different discretisations. The accumulator is a component of an integrated modelling framework and can be completed by the modeller with custom operations for spatial and temporal scaling. To illustrate the applicability of the accumulators an integrated model is developed that couples an existing land use change model and hydrological component models at different spatial and temporal scales. The accumulators as building blocks allow modellers to construct multi-scale integrated models in a flexible manner.

Key words | hydrology, integrated modelling, interdisciplinary modelling, land use change, Python, spatio-temporal scaling

Oliver Schmitz (corresponding author)
Derek Karssenberg
Department of Physical Geography,
Faculty of Geosciences,
Utrecht University,
Heidelberglaan 2, PO Box 80115, 3508 TC Utrecht,
The Netherlands
E-mail: *o.schmitz@uu.nl*

Oliver Schmitz
Elga Salvadore
Lien Poelmans
Flemish Institute for Technological Research (VITO),
Unit Environmental Modelling,
Boeretang 200, 2400 Mol,
Belgium

Elga Salvadore
Vrije Universiteit Brussel,
Department of Hydrology and Hydraulic Engineering,
Brussels,
Belgium

Johannes van der Kwast
UNESCO-IHE Institute for Water Education,
Westvest 7, PO Box 3015, 2601 DA Delft,
The Netherlands

INTRODUCTION

Integrated models simulating spatio-temporal changes are important tools for exploring characteristics and interactions of human–natural systems. These models represent real world processes by spatial state transition functions, and provide insight into processes and interactions between system components, which may lead to an improved scientific understanding (e.g., Rotmans 1990; Liu *et al*. 2002). In hydrology, integrated models are widely used for river basin management and risk assessment (e.g., Abbott *et al*. 1986a, 1986b; Jakeman & Letcher 2003; Ahrends *et al*. 2008; Argent *et al*. 2009; de Kok *et al*. 2009; Karaouzas *et al*. 2009; de Roo *et al*. 2011). These models combine knowledge from various domains, such as hydrology, ecology and agricultural sciences (e.g., van Ittersum *et al*. 2008; de Kok *et al*. 2009; Roberts *et al*. 2010; Janssen *et al*. 2011; Harvey *et al*. 2012). The construction of integrated models requires a scientific, semantic and technical integration of individual models (Argent 2004; Hinkel 2009; Janssen *et al*. 2011; Knapen *et al*. 2013; Elag & Goodall 2013). Modular construction approaches are valuable in the development cycle of integrated models, as shown by several authors (e.g., Liu *et al*. 2002; Voinov *et al*. 2004; Argent 2005; McIntosh *et al*. 2005; Papajorgji 2005; Rizzoli *et al*. 2008; Castronova & Goodall 2010; Holzworth *et al*. 2010; Peckham

doi: 10.2166/hydro.2013.180

et al. 2013; Granell et al. 2013a). These studies recommend that environmental modellers describe particular environmental, social or economic processes in confined component models, and that these component models are coupled to interdisciplinary, integrated models.

Programming languages and several software packages exist that support a modeller in the construction of component models and their integration (see e.g., Steiniger & Hay 2009; Jagers 2010; Granell et al. 2013b). Traditional system programming languages such as Fortran, C or C++ allow for the implementation of virtually any integrated model application from scratch or by combining software libraries. Alternatively, component models can be built by using domain specific modelling languages such as SimuMap (Pullar 2004), PCRaster (Wesseling et al. 1996), Ocelet (Degenne et al. 2009) or SITE (Schweitzer et al. 2011). These languages are specifically designed for the development of spatio-temporal models in hydrology or other domains. They provide high-level operations on spatio-temporal data types that implement algorithms for common environmental processes, for example, the kinematic wave equation (Chow et al. 1988). Moreover, they provide domain specialists with a comprehensible syntax for model building. Some domain specific languages such as Simile (Muetzelfeldt & Massheder 2003), STELLA (2013) or ExtendSim (2013) come with graphical modelling languages. Software tools with an emphasis on model coupling such as the Open Modelling Interface (OpenMI, e.g., Moore & Tindall 2005; Gregersen et al. 2007), the Typed Data Transfer library (TDT, Hinkel 2009), the Bespoke Framework Generator (Armstrong et al. 2009) or the Model Coupling Toolkit (Warner et al. 2008) provide communication protocols that standardise component interactions. Environmental modelling frameworks such as the Earth System Modelling Framework (e.g., Hill et al. 2004), the Community Surface Dynamics Modeling System (e.g., Peckham et al. 2013), E2 (Argent et al. 2009) or OMS3 (David et al. 2013) provide data types and operations for the construction of model constructs and functionality for their coupling. Finally, software packages that support additional tasks such as data retrieval, data analysis or visualisation (e.g., Hunter 2007; Kiehle et al. 2007; Huang et al. 2011; R Development Core Team 2013; Werner et al. 2013) might need to be integrated in the model development as well.

These software packages can assist modellers both in the technical development phase of individual component models and in their integration. In addition to the problem of choosing the appropriate software tools and the technical challenge of model implementation itself, modellers face a conceptual challenge at the coupling phase as well: given that each individual model has its own specific spatial and temporal discretisations, also called resolution or support size, coupling will result in complex multi-scale integrated models. Modellers are therefore required to adjust the variables that are exchanged between component models such that these variables conform to both the component model providing the variable and the one receiving the variable. These adjustments can be related to either the spatial discretisation, the temporal discretisation, or both. Adjusting the spatial discretisation is required, for example, when coupling a continental scale coarse grid climate model to a hydrological catchment model that uses smaller grid cells. Similarly, adjusting the temporal discretisation is required when coupling component models that use different time steps. This adjustment of temporal discretisations between component models is a common task in hydrology, where for example highly dynamic processes such as surface runoff are coupled to slower processes such as groundwater flow. An appropriate coupling therefore needs spatial and temporal aggregation or disaggregation of the exchanged variables.

Conceptual methodologies to bridge discrepancies between component models are available as in, for example, common spatial scaling techniques, or methodologies describing temporal aggregation or disaggregation (e.g., Blöschl & Sivapalan 1995; Bierkens et al. 2000; Rastetter et al. 2003; Skøien et al. 2003; López et al. 2005; Vermaat et al. 2005; Karssenberg 2006; Barnes et al. 2007; Malone et al. 2012). However, using multiple software packages for the construction of individual component models, and their coupling at various spatial and temporal scales, remains a tedious task for a model builder. A modeller, however, can use preliminary alternatives to implement scaling operations. Implementing scaling concepts for dynamic models often remains a manual exercise when using system programming languages. The TDT (Hinkel 2009), for example, provides support for the transfer of data structures between different programming languages but it does

not provide building blocks for upscaling or downscaling these data structures. OpenMI (Moore & Tindall 2005) provides the model developer with a construct to buffer component outputs and to use these further on with different interpolation techniques (e.g., Elag *et al.* 2011). The OpenMI framework, however, was designed to couple existing models, and support for the construction of new component models from scratch is thus limited. Other frameworks tailored to the development of integrated models, such as the CSDMS (e.g., Peckham *et al.* 2013) or ESMF (e.g., Hill *et al.* 2004) provide coupler components for spatial and temporal interpolation or extrapolation. The implementation of these scaling operations is done on the level of system programming languages. Visual modelling languages such as STELLA (2013) or ExtendSim (2013) provide built-in building blocks for upscaling or downscaling. However, these scaling operations can only be used for lumped models since these software packages have limited intrinsic support for spatial modelling. Spatial explicit operations need to be added using other packages, as was shown by van Deursen & Maes (2008) and Voinov *et al.* (2004). Stasch *et al.* (2012) focused on the development of aggregation services for observed data but do not explicitly support runtime aspects of dynamic models. Mennis (2010) proposed a multidimensional map algebra but provided a limited set of built-in operations for spatio-temporal aggregation. Integrated modelling frameworks providing support for both the construction of individual component models and for the implementation of scaling techniques in a suitable way for domain specialists, such as hydrologists or ecologists, are rare.

This paper describes such a programmable building block for the description of scaling operations, performing aggregation or disaggregation on spatial and temporal discretisations required in component-based model setups. This is done by including an accumulator building block in a process-based spatio-temporal modelling framework prototype (Schmitz *et al.* 2013). This accumulator enables a model builder to describe algorithms on two-dimensional raster data with conventional map algebra operations (Tomlin 1990), and to describe generic or custom aggregation or disaggregation operations on the temporal discretisation. First, the concepts of the component-based software development practice and its application in hydrological and environmental modelling are briefly introduced. At the same time, the issue of spatio-temporal misalignment between model building blocks occurring at model runtime is raised. Next, the accumulator concept to adjust couplings is shown, and its technical implementation within the modelling framework is outlined. Finally, the application of the accumulator in bridging spatio-temporal discrepancies in hydrological models is illustrated. We report on a first phase of a flood-risk assessment case study for a catchment in Flanders, Belgium. An existing, external land use change model is coupled to a reimplemented spatially distributed hydrological model. The purpose of this case study is to demonstrate the technical implementation of the integrated model, and not to provide model analysis. We conclude with a discussion on noteworthy items in the coupling of multi-scale component models.

BRIDGING SPATIO-TEMPORAL DISCREPANCIES IN MODULAR MODEL SETUPS

The software engineering and integrated modelling disciplines face comparable tasks in the construction of their applications. Both domains involve the construction of complex systems, and they approach this task by assembling modular, reusable and reliable components into larger systems. This construction process is guided by software engineering practices that organise the structured development of complex software products. Examples are the spiral development method (e.g., Boehm 1988), extreme programming (e.g., Beck & Andres 2004), or the more recent agile development strategies (e.g., Cohen *et al.* 2004; Kniberg 2011; Rubin 2012). The objective of these practices is to improve the software development process and product quality. This is achieved by defining tasks and sequencing of software specification, design, implementation and application. The need to adopt structured development practices in the construction of integrated models is increasingly recognised (e.g., Scheller *et al.* 2010; Schmolke *et al.* 2010; Verweij *et al.* 2010). We now briefly introduce the component-based software engineering practice for the construction of modular hydrological and environmental components. Afterwards, we show its limitations in the

coupling of spatio-temporal component models and provide an approach to resolve these limitations.

Component-based construction of integrated models

Component-based software engineering promotes the construction of larger systems out of reusable, independently developed components of confined functionality (e.g., Szyperski 2002). Independent components can be obtained by the concept of encapsulation where associated functionality is grouped within each component. Components also need to be developed in a decoupled fashion. This implies that no references to other components should be specified in a component to avoid external dependencies. Component-based practices reduce the complexity of larger systems, and enhance maintenance and reliability of modules. Adopting these practices in environmental modelling can lead to the development of generic, reusable components that are deployable in various model applications (e.g., Mineter *et al.* 2003; Argent *et al.* 2006; Hinkel 2009; Donchyts & Jagers 2010; Voinov & Cerco 2010). By defining standardised interfaces for the components, and by standardising the exchange of variables, developers can use these components to assemble larger systems in a flexible way. Figure 1(a) shows a conceptual component-based setup of an integrated model. The process representations are encapsulated and different for each component model. The interactions between the component models and the exchange of variables follow a standardised protocol.

Component-based development practices provide approaches for the technical integration of environmental models. Modellers following these practices can map conceptual subsystems into corresponding software components, and they can couple these component models to integrated systems. However, individual component models holding dynamic environmental processes are often associated with characteristic spatial and temporal discretisations (see e.g., Blöschl & Sivapalan 1995; Skøien *et al.* 2003). When coupling component models with different temporal or spatial characteristics, modellers are confronted to resolve the misalignment (Figure 1(b)). A direct exchange of the variables is not feasible. Modellers therefore need to specify methods for temporal upscaling

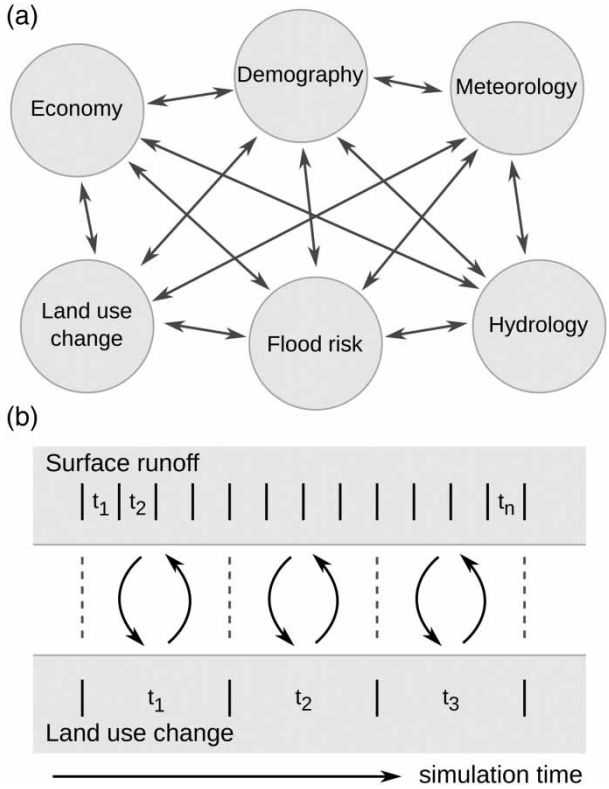

Figure 1 | Structural and dynamic characteristics of integrated models following component-based development approaches. (a) Seamless construction of integrated models enabled by standardised interfaces and a standardised communication protocol. (b) Online coupling of component models with different time steps require additional measures to adjust exchanged variables. t_i denote time steps, dashed lines denote exchange periods.

or downscaling of model variables or parameters to couple the component models.

A potential approach to resolve spatial and temporal discrepancies between component models is to modify the internal process representation. For example, the time step of one model could be adapted such that it matches the temporal discretisation of the coupled counterpart (e.g., Monninkhoff & Kernbach 2006). This approach is not favourable because of several reasons. First of all, a modification of encapsulated process descriptions places external dependencies on the working of a component model. Adding particular dependencies is against the principle of individual executable components, and results in a less generic applicability of models. Second, a process description can be restricted by spatio-temporal constraints of the used numerical solution scheme, or by physically meaningful characteristic spatio-temporal scales. A modification of such processes is therefore

undesired. Finally, a modification of process descriptions may not be possible if these are compiled into executable applications. Consequently, aggregation or disaggregation of model variables appears to be the best approach.

Software engineering suggests the adapter or wrapper pattern (e.g., Gamma *et al.* 1995) to bridge misaligned components without changing the internals of the modules. Thereby, additional code is added that transforms characteristics of output interfaces into input interfaces. The adapter pattern is a commonly used approach in the development of environmental models and in modelling frameworks (e.g., Joppich & Kürschner 2006; Gregersen *et al.* 2007; Villa 2007; Schmitz *et al.* 2009; Castronova & Goodall 2010; Becker & Schüttrumpf 2011; Bulatewicz *et al.* 2013; OpenMI 2 SDK 2013). We extend the adapter pattern to an accumulator building block suitable to bridge component models regarding the spatial and temporal discretisation.

Adjusting spatio-temporal discrepancies

The accumulator is used to spatially or temporally aggregate (upscaling) or disaggregate (downscaling) exchanged variables at model runtime. The accumulator accesses the interfaces of the coupled component models to obtain outputs and to provide adjusted variables, and performs a specific scaling operation implemented by the modeller. To be applicable as a generic model building block, an accumulator needs to implement functionality for different coupling situations. First, the accumulator needs to be able to adjust the spatial discretisation of the exchanged variables. Second, the accumulator should support the exchange of variables between coupled models in both directions. For a coupling of components with smaller time steps to components with larger time steps, a temporal aggregation of variables is required. The contrary direction requires a disaggregation of variables.

Figure 2 shows the situation of temporal misalignment between two component models, and introduces the notation that is used in the remainder of the paper. A component model C_i holds descriptions representing dynamic environmental processes. These dynamic processes are simulated by iterating a state transition function over a set of discrete time steps Δt_i (Beck *et al.* 1993; Burrough 1998; Karssenberg & de Jong 2005a). Interactions between two components within a time step, as for example in solving differential equations, cannot be represented. The component models are therefore coupled externally (Morita & Yen 2002).

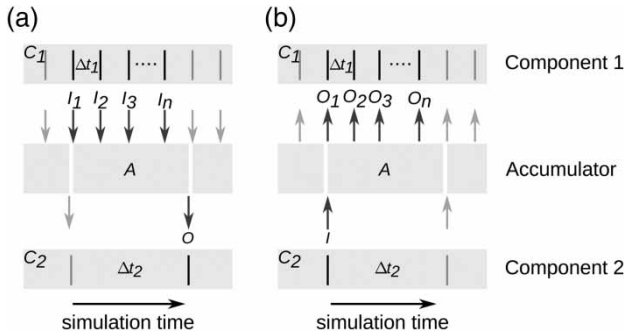

Figure 2 | Input and output variables considered at an exchange period by an accumulator *A*. (a) Temporal aggregation and (b) temporal disaggregation. Each execution of *A* is associated with an exchange period consisting of all time indices for input variables I_t and output variables O_t restrained by Δt_2.

The exchanged variables represent spatially distributed attributes such as land use classes or hydrological characteristics. The framework uses two-dimensional raster maps with a regular grid as spatial discretisation. The component models provide these maps as output variables for each of these time steps.

Figure 2(a) shows the exchange from component model C_1 with a smaller time step Δt_1 to component model C_2 with larger time step Δt_2. This direction of the exchange requires an aggregation of the variables from C_1 before they are in an acceptable input format for C_2. The accumulator needs to obtain a set of $I_1, \ldots, I_n$ maps of a variable from C_1, and should provide one output map O to C_2. The modelling framework determines the interval that needs to be considered for the input maps by means of the time step Δt_2 of the component model C_2. The number n of obtained input maps is determined by the time step Δt_1 of C_1.

Figure 2(b) shows the opposite direction of exchange. In this case, a disaggregation of a variable from component model C_2 with time step Δt_2 to component model C_1 with smaller time step Δt_1 is required. The accumulator needs to obtain one input map I from C_2, and to provide a set $O_1, \ldots, O_n$ output maps in accordance to the time step Δt_1 of C_1.

The accumulator for aggregation

We now formalise the accumulator function such that aggregation and disaggregation can be specified in order to couple

component models that run at a different spatial or temporal discretisation. As we focus on the development of integrated models for dynamic environmental systems, we will illustrate the construction of the accumulators based on maps with two-dimensional raster data. Comparable concepts will apply to other variable types such as one- or three-dimensional data types.

We first consider an accumulator coupling a component model C_1 with a smaller time step Δt_1 to a component model C_2 with a larger time step (see Figure 3). For example, a surface runoff component with a daily time step needs to be coupled to a groundwater flow process component with a monthly time step. Spatially distributed recharge values are output variables of the component model C_1 and are required as input by C_2. To guarantee a flawless data exchange between the building blocks, the modeller needs to be able to adjust the spatial discretisations of maps as well as the temporal discretisation. Adjusting the temporal discretisation requires an aggregation of the recharge values for each month.

To obtain the functional description of an accumulator, we first define f (see Figure 3) describing the scaling operation on the spatial discretisation:

$$f\colon f(I_1, \ldots, I_n) \mapsto O_1, \ldots, O_n \tag{1}$$

In Equation (1), f describes the spatial aggregation or disaggregation of the exchanged maps, for example, by calculating total or average values for each subcatchment, or by rescaling the spatial discretisation of a raster map. The transformation is performed on each of the individual input map I_t and results in n temporary output maps. For a time step of the groundwater component, for example, in the month of January, n corresponds to 31 daily recharge values.

Furthermore, we define a temporal operation g as

$$g\colon g(I_1, \ldots, I_n) \mapsto O \tag{2}$$

The operation g takes a set of n input maps I_t and aggregates these into one output map O. For temporal operations on raster maps, g is executed for each cell location. Examples for aggregating operations g are logical operations indicating whether a location was affected by

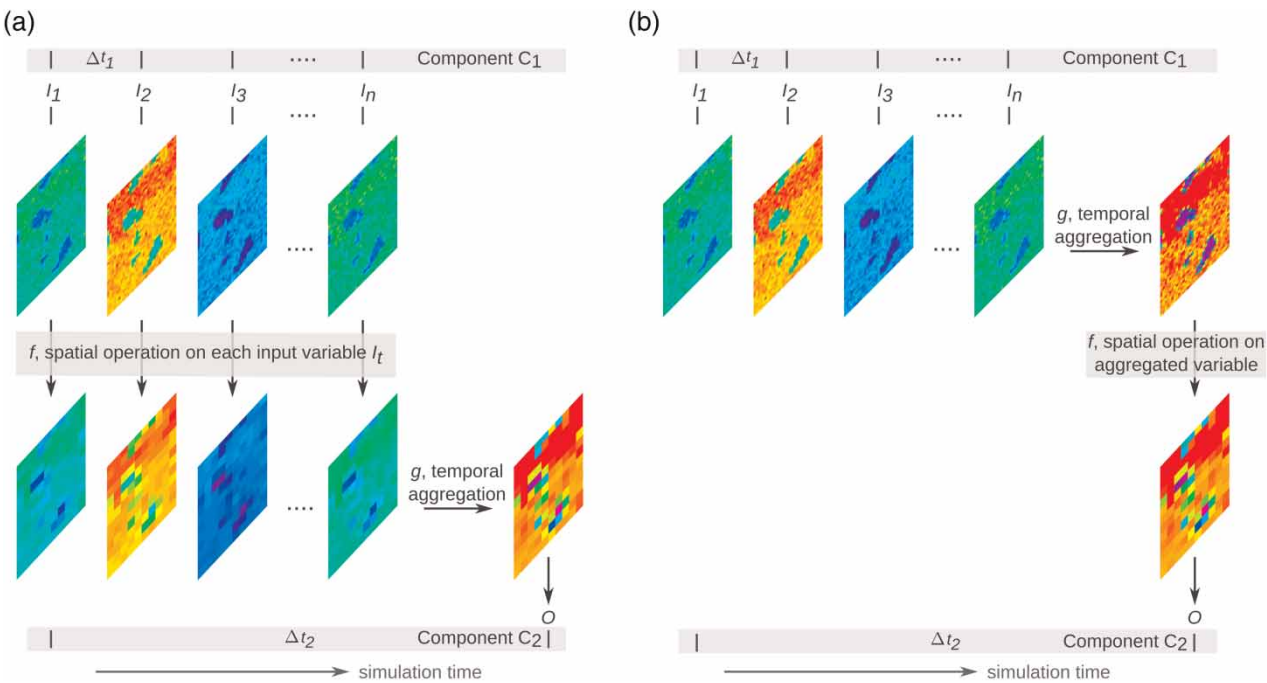

Figure 3 | Accumulator A (see Figure 2(a)) aggregating two-dimensional variables obtained from a component model C_1 with time step Δt_1. (a) The input maps $I_1, \ldots, I_n$ are first treated spatially (f) and then aggregated temporally (g) to obtain the output O, which is further used by C_2. (b) First executes g, then f. Note that the aggregation (i.e., calculation of O) is done for each time step Δt_2 of C_2.

phenomena such as flooding or fire, or statistical operations calculating mean discharge values.

With f describing the spatial scaling operations and g describing the temporal scaling operations, the aggregating operation of an accumulator can be described. The accumulator obtains a set of spatio-temporal input maps $I_1, \ldots, I_n$ from component model C_i, and provides one aggregated map O to C_j according to the execution of the operation A

$$A = \begin{cases} g \circ f\colon g(f(I_1, \ldots, I_n)) \mapsto O \\ f \circ g\colon f(g(I_1, \ldots, I_n)) \mapsto O \end{cases} \quad (3)$$

The aggregated output map O is obtained as a result of a composition of the spatial operation f and the temporal operation g. The modeller can specify the aggregation of input map by two different paths. In the first case described in Equation (3), the operations on the spatial discretisation are performed first and then the maps are aggregated over time (Figure 3(a)). In the second case, the aggregation over time is performed first, and afterwards the spatial operation is executed (Figure 3(b)).

The accumulator for disaggregation

The second purpose of the accumulator is to disaggregate variables from larger time steps to smaller time steps. An example of such an application in hydrology is the disaggregation of precipitation data that are only available in a lower resolution than required for high-resolution processes such as surface runoff. A disaggregation is therefore required to cover short-term intensity effects (e.g., Güntner *et al.* 2001). The corresponding accumulator is shown in Figure 4, where maps are transferred from component model C_2 with larger time step Δt_2 to a component model C_1 with a smaller time step Δt_1. When disaggregating an output map from a component with a monthly time step to a component with daily time step, for example for the month January, maps for 31 time steps need to be calculated.

Comparable to the aggregation of maps, the modeller can select from two paths to disaggregate a map from C_2 to a set of output maps for C_1. The first option is to expand the temporal discretisation to match the smaller time step by copying the input variable, and then to perform a spatial operation to modify each map of the expanded set

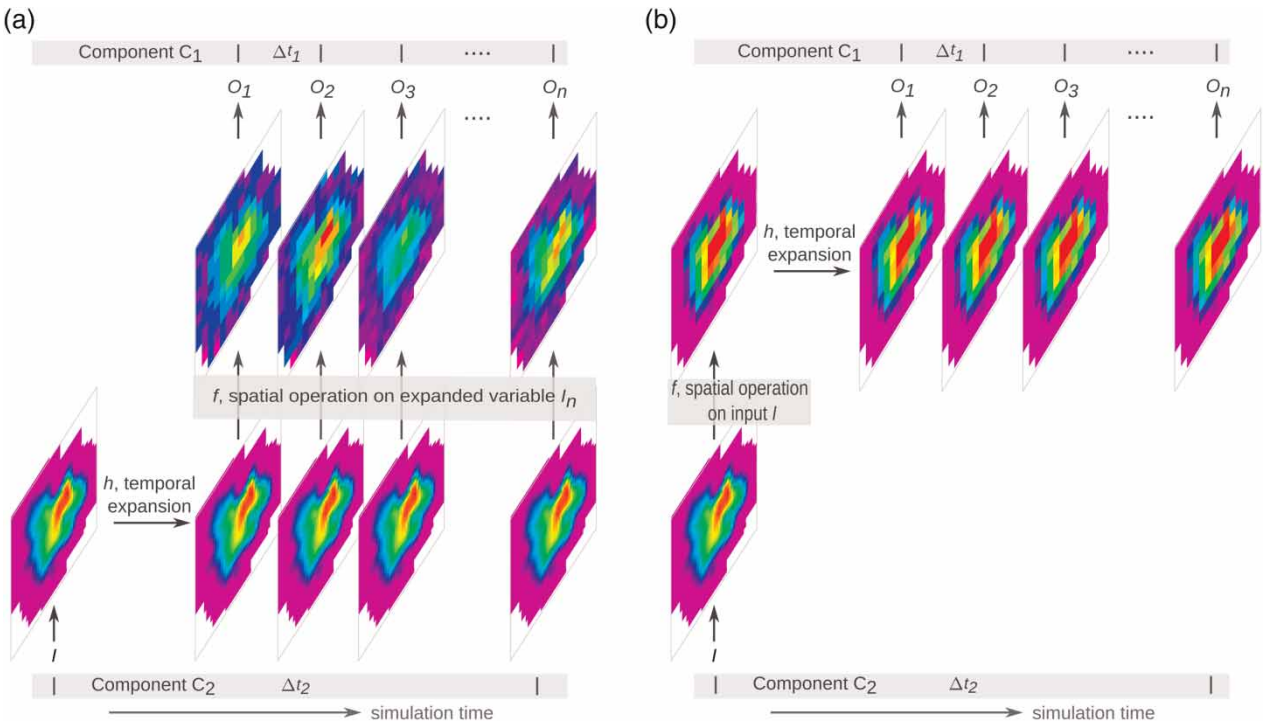

Figure 4 | Accumulator A disaggregating one input map I from component C_2 to a set of output maps O_i in agreement with the time steps Δt_1 of C_1 (see Figure 2(b)). (a) First executes the temporal operation h, then the spatial operation f. (b) First executes f, then h. The disaggregation is done for each time step Δt_2 of C_2.

(Figure 4(a)). The second option is to first execute the spatial operation, and then to copy this modified map an adequate number of times appropriate for the temporal discretisation of C_1 (Figure 4(b)).

For the operation f modifying the spatial discretisation, we use the same description as given in Equation (1). We define an operation h on the temporal dimension expanding one input map I to a set of n output maps O_i according to

$$h\colon h(I) \mapsto O_1, \ldots, O_n \tag{4}$$

We now can define the function A of an accumulator for disaggregation as

$$A = \begin{cases} h \circ f\colon h(f(I)) \mapsto O_1, \ldots, O_n \\ f \circ h\colon f(h(I)) \mapsto O_1, \ldots, O_n \end{cases} \tag{5}$$

IMPLEMENTING THE ACCUMULATORS

The concept of a generic accumulator is implemented in a prototype version of an integrated modelling framework. We briefly introduce the framework, and show how a modeller can use the provided accumulator templates to implement operations for spatio-temporal aggregation or disaggregation, respectively.

A modelling framework for component model construction and coupling

The environmental modelling framework introduced in Schmitz *et al.* (2013) is used to demonstrate the accumulator concept. The framework provides one environment for the construction of process-based spatio-temporal component models and their coupling. The modelling framework follows component-based software development practices (e.g., Szyperski 2002; Rizzoli *et al.* 2008; Argent *et al.* 2009) and provides component templates with standardised input and output interfaces. The modeller can complete component model templates with descriptions representing natural processes at individual spatial and temporal discretisations. These process representations can be specified by operations building upon the map algebra concept (e.g., Tomlin 1990; Karssenberg *et al.* 2007). Accumulator templates can be completed with map algebra operations specifying spatial and temporal scaling. The modelling framework will call the update methods of components and accumulators and hence execute the operations specified by the modeller. The framework includes a management layer that schedules the execution of component models and accumulators. A central instance of the modelling framework maintains a shared time line between all components and derives the execution order based on the current time steps of the model components. The execution of the accumulators is scheduled immediately before a component proceeds to the next time step.

The modelling framework is tailored to domain specialists such as hydrologists or ecologists, who do not necessarily have explicit knowledge in traditional system programming languages. It uses the general-purpose scripting language Python (2013) as model implementation environment. Spatio-temporal operations and data types following the map algebra concept (Karssenberg *et al.* 2007) and the framework for integrated modelling can be imported as Python modules. If necessary, additional modules for modelling purposes such as libraries for numerical arrays or geospatial data formats can be imported as well (e.g., Langtangen 2007; Van Der Walt *et al.* 2011; GDAL Development Team 2013). By extending the standard Python language with support for spatio-temporal modelling, a modeller can use a flexible modelling environment for the construction of integrated models.

Technical implementation

The accumulator base class

The accumulator is a generic building block that modellers can use to describe spatial or temporal aggregation or disaggregation operations. For this purpose, the framework provides a base class shown in the Unified Modelling Language (Booch *et al.* 2005) diagram in Figure 5. The base class provides public methods that correspond to the functionality introduced in the previous section. The model builder can complete the following two methods. The *time_aggregate* method corresponds to g (Equation (2)) and performs the aggregating operations. The

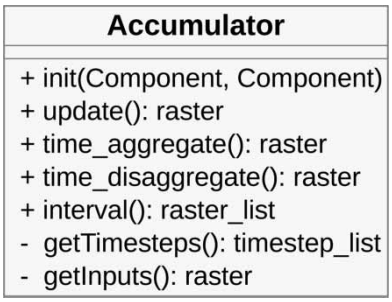

Figure 5 | UML diagram of the accumulator class provided by the modelling framework.

time_disaggregate method corresponds to h (Equation (4)) and holds disaggregating operations. The *interval* method enables a modeller to loop over a sequence of spatial variables, and therefore to modify each of the inputs $I_1, \ldots, I_n$ or outputs $O_1, \ldots, O_n$ respectively. The order of the spatial and temporal operations can be specified in the *update* method, corresponding to A (Equations (3) and (5)). The remaining methods shown in Figure 5 are used by the modelling framework to obtain the time step information of the component models and the corresponding spatial variables at model runtime.

Obtaining spatial variables at model runtime

At model construction, a modeller specifies in the accumulator how spatial and temporal discretisation differences will be bridged. By calculating the sum or average, for example, a daily variable could be aggregated to a monthly value. The modelling framework executes this accumulator repeatedly at model runtime. It updates the input and output time steps that need to be considered (see Figure 2) and therefore organises the correct progress of the accumulator in time.

The accumulator is instantiated with the *init* method. The two arguments are references to the providing component model and the receiving component model, respectively. At model runtime, the accumulator receives the current time step t from the modelling framework and determines the input and output intervals of the coupled component models. In the case of aggregation, the accumulator queries the receiving component model with the *getTimeSteps* method, and obtains the current time step t and the previous time step $t-1$. These time steps are used as interval boundaries to query the providing component model about all available time steps in this interval. For these time steps, the *getInputs* method obtains the spatial variables from the providing component model, and the modeller can write custom code to loop through the spatial variables with the *interval* method. In the case of disaggregation, the time steps t and $t-1$ of the providing component determine the output interval.

Implementing aggregating operations

To illustrate how a modeller can implement the accumulators, we assume a coupling of a rainfall–runoff model with a daily time step to a groundwater model with a monthly time step. Infiltration values need to be transferred from the rainfall–runoff model to the groundwater model. The accumulator provides access to the incoming maps $I_1, \ldots, I_n$, and needs to provide one outgoing map O (see Figure 2(a)). The number of time steps n equals in this scenario the number of days in each month.

To describe aggregating operations, the modeller needs to implement the *time_aggregate* and update methods. Figure 6 shows the implementation of the aggregating accumulator. In the *time_aggregate* method, the modeller specifies the operation g (Equation (2)) as custom code. Here, the sum of the infiltration values over 1 month is calculated for each cell. Line 7 describes the iteration over the daily input maps $I_1, \ldots, I_n$. Lines 8 and 9 describe the operations that are performed on each individual map I_i. First, a spatial operation scale (f in Equation (1)) is rescaling the infiltration map of the current time step to match the

```
1 class TotalInfiltration(Accumulator):
2   def __init__(self, inComponent, outComponent):
3     Accumulator.__init__(inComponent, outComponent)
4
5   def time_aggregate(self):
6     sum = scalar(0)                                  ┐
7     for inf in self.interval():                      │
8       dailyInfiltration = self.scale(inf)            │ Custom code
9       sum = sum + dailyInfiltration                  │
10    return sum                                       ┘
11
12  def update(self):                                  ┐
13    return self.time_aggregate()                     ┘ Custom code
```

Figure 6 | Python script showing an accumulator stub. The custom code is implemented by a modeller and describes an aggregating operation.

spatial discretisation of the groundwater model (line 8). Next, the result is added to a map holding the monthly sum. The modeller specifies the accumulator as aggregating accumulator in the *update* method (line 13).

Implementing disaggregating operations

A further application case of the accumulator is to perform disaggregating operations, which are required when coupling component models with larger time step to components with smaller time steps (see Figure 2(b)). The accumulator enables a modeller to access the incoming map I, and needs to provide a set of output maps $O_1, \ldots, O_n$.

Figure 7 shows the implementation of the disaggregating accumulator. The modeller needs to implement the *time_disaggregate* (line 5, h in Equation (4)) and the *update* method. Again, the *interval* method (line 6) enables a modeller to loop over the time steps, in this case modifying the output time steps. The *normal* operation adds random noise to each cell of the incoming map, which is taken here as an example of a simple downscaling technique. Additional interpolation techniques that can be implemented by a modeller in the *time_disaggregate* method can be found, for example, in Bierkens *et al.* (2000) and Elag *et al.* (2011). The class returns the set of output maps with the call of *time_disaggregate* in the *update* method.

APPLICATION OF THE MODELLING FRAMEWORK

To illustrate the usability of the accumulators in the component-based development process of integrated models, we present an instructive proof-of-concept model application for a Belgian catchment. The study focuses on the

```
1 class RandomNoise(Accumulator):
2   def __init__(self, inComponent, outComponent):
3     Accumulator.__init__(inComponent, outComponent)
4
5   def time_disaggregate(self):
6     for variable in self.interval():      ]
7       variable = variable + normal()      ]  Custom code
8
9   def update(self):
10    return self.time_disaggregate()       ]  Custom code
```

Figure 7 | Python script showing an accumulator performing a disaggregating operation.

technical aspects in the coupling of multi-scale component models. The correct execution order of the components and the data exchange at appropriate time steps have been verified (see e.g., Oreskes *et al.* 1994; Sargent 2013). Model evaluation procedures such as calibration and uncertainty analysis (e.g., Refsgaard *et al.* 2007; Hall & Solomatine 2008) have not yet been carried out.

A land use change model was coupled to a hydrological model using an existing land use change model for Flanders, based on the MOLAND modelling framework (Barredo *et al.* 2004; Engelen *et al.* 2007), and a set of hydrological component models based on the distributed rainfall–runoff model WetSpa (e.g., Wang *et al.* 1996; Liu *et al.* 2003).

The integrated model was applied to the Kleine Nete catchment, a 581 km^2 sub-catchment of the Scheldt basin. The catchment is located in the northeastern part of Flanders, Belgium (see Figure 8). Elevation ranges between 3 and 48 metres, and the climate is temperate with an average precipitation of 830 mm. The catchment is classified as 60% agricultural area, 20% as forests and 10% as urbanised area. The average population density is around 380 inhabitants per km^2 (Dams *et al.* 2012, 2013).

Model structure

The conceptual setup of the model with the main building blocks and interactions is shown in Figure 9. Component models are used to model the land use change domain and the hydrological domain. Land use change and hydrology are coupled to data providers supplying population data as time series input, and climate data represented by a set of maps with spatially distributed precipitation and potential evapotranspiration values, respectively.

We choose these two domains for our application, as the evaluation of feedback mechanisms between land use change systems and hydrological systems is relevant and can be used to better analyse and forecast the interactions between land use change and hydrology, in particular to predict and assess impacts of flooding. Feedback mechanisms between land use change and hydrology may lead to an amplification or attenuation of changes. An explicit inclusion of feedback mechanisms is, however, still one

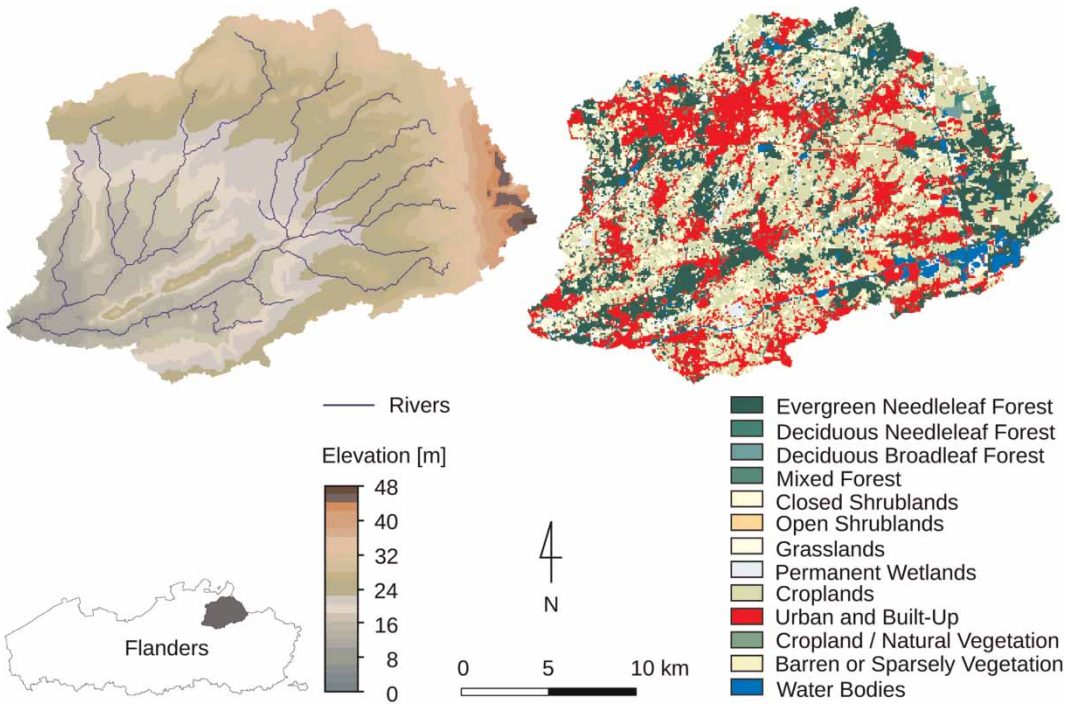

Figure 8 | Location, topography, river network and land use of the Kleine Nete catchment.

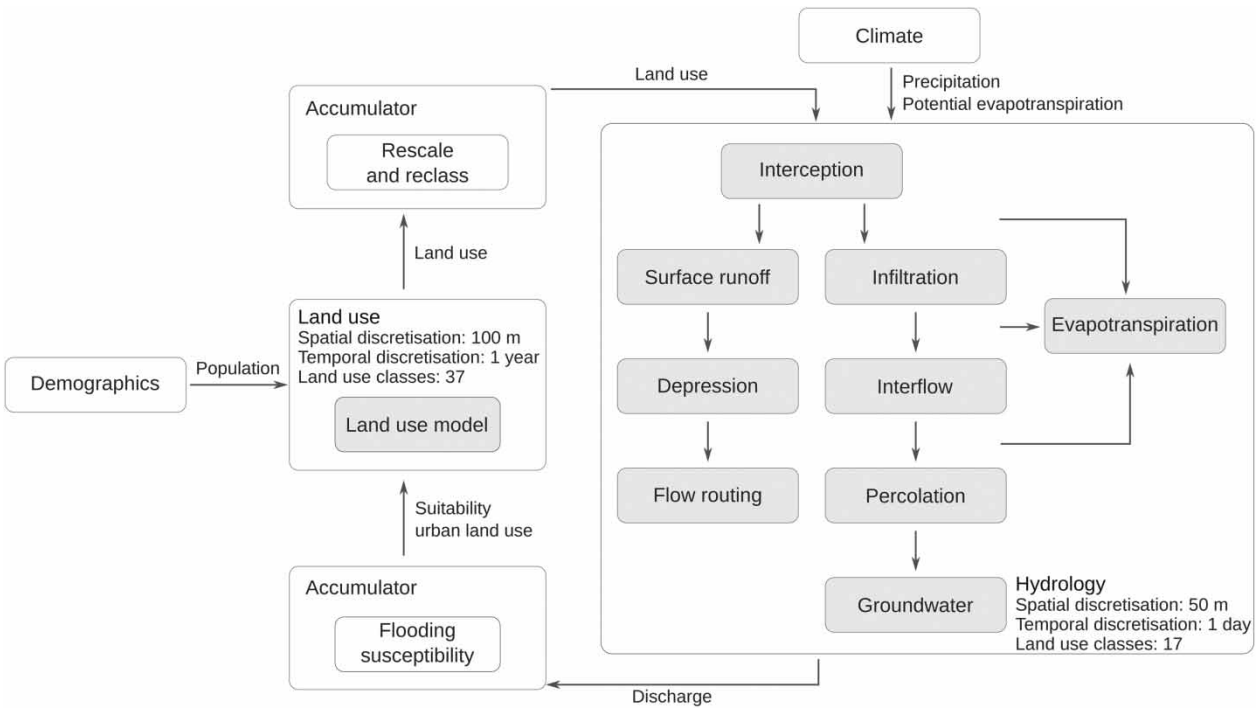

Figure 9 | Conceptual diagram showing the interactions between the land use change and hydrological component models.

of the major challenges in studying land use systems (Claessens *et al.* 2009). In this application, feedback between the component models is modelled by an exchange of land use maps and flood zone maps. In this way, we evaluate how an increase in sealed surface areas simulated by the land use change model influences surface runoff, and how this affects the risk of flooding. Consecutively, this change in flood risk is used as an input to the land use change model calculating land use change for the next time step taking into account the updated flood risk map.

The land use change component

In this study, a simplified version of the 'RuimteModel' land use model for Flanders is used. RuimteModel is an advanced application of the MOLAND modelling framework (Barredo *et al.* 2003, 2004; Hagen-Zanker *et al.* 2005; Engelen *et al.* 2007) and has been developed to support spatial planners and policy-makers in Flanders (Engelen *et al.* 2005, 2011; de Kok *et al.* 2012). It is a constrained cellular automata land use model that simulates the likely future development of 37 land use types with a temporal discretisation of 1 year and a spatial discretisation of one hectare, given alternative planning and policy scenarios and socio-economic trends.

The model input consists of five geographic information system (GIS) datasets: the actual land use types, the accessibility to the transport network, the physical suitability for different land use types, the zoning status of the land, and a number of socio-economic characteristics such as population growth and employment in the area. Based on the local interactions with neighbouring land use types, accessibility, physical suitability and administrative zoning, a transition potential is calculated for each land use type. The land use type with the highest potential is assigned to each of the cells in the study area until the regionally imposed land use demands are met.

For the test case of the Kleine Nete, all socio-economic trends, model parameters and GIS layers, except for the physical suitability, were adopted from the Business-As-Usual (BAU) scenario of the RuimteModel (Engelen *et al.* 2011). The BAU scenario describes the land use dynamics in Flanders between 2010 and 2050 under the prevailing policies and socio-economic trends. The physical suitability for land use types, on the other hand, is based on the output of the hydrological model and will be updated for each 1-year time step of the land use model.

The hydrological components

The WetSpa model, reimplemented in Python by the authors, is used to represent the hydrological processes in the Kleine Nete catchment. The WetSpa model is a spatially distributed rainfall–runoff model for simulations at the catchment scale (e.g., Batelaan *et al.* 1996; De Smedt *et al.* 2000; Liu *et al.* 2003; Safari *et al.* 2012). The model simulates several physical processes at the level of individual raster cells, such as interception, depression storage, evapotranspiration, runoff, interflow and groundwater recharge (see Figure 9). The main physically based equations of the model are described in more detail by Liu & De Smedt (2005). Surface runoff is produced using a modified coefficient method based on physical characteristics of each raster cell, i.e., topography, soil type, land use, soil moisture and rainfall intensity. This method calculates a cell potential rainfall excess coefficient or potential runoff coefficient, having a value between 0 and 1. The coefficient acts as a multiplier of the net precipitation to separate the surface runoff component from the infiltration component. Values of this coefficient are interpolated from the literature and adapted to the real physical characteristics of the basin. The calculated runoff is then routed as overland flow and channel flow using the linear diffusive wave approximation method. Interflow is computed based on Darcy's law and the kinematic wave approximation as a function of the effective hydraulic conductivity and the hydraulic gradient. The groundwater flow is estimated with the linear reservoir method on small subcatchment scale as a function of groundwater storage and a recession coefficient. A modeller can select the time step discretisation of inputs and outputs from a range of 1 hour, days, months or years.

The inputs of the model are: precipitation and potential evapotranspiration time series, which are distributed over the catchment with the Thiessen polygons method if not available in a distributed way such as radar data. Spatially distributed parameters are derived from three base maps using GIS software: topography, soil texture and land use. Outputs of the model are flow hydrographs at the catchment and subcatchment outlets, and maps of evapotranspiration,

soil moisture, interception, surface runoff, groundwater recharge and interflow for each time step or selected periods.

The Python version of the WetSpa model maintains the same capabilities as the original Fortran implementation. However, a different model structure was developed and different GIS software (PCRaster 2013) was used. The new structure is modular, allows land use to be received as a dynamic input variable, and every physically based process is coded in a separate component model. These component models are coupled and exchange data at runtime through the modelling framework prototype (see Figure 9). Existing case studies were used to verify the correctness of the reimplementation.

For the test case of the Kleine Nete, the topography, resampled to 50 m from the original 25 m resolution digital elevation model of Flanders (OC GIS-Vlaanderen 2001), is used as GIS input map. Meteorological data of rainfall and potential evapotranspiration were obtained from the Royal Meteorological Institute of Belgium. The temporal discretisation was set to a 1-day time step.

Flood risk assessment

To model the feedback from the hydrological component models, we incorporate the effects of the stream discharge in the physical suitability for urban land use classes. We modified the input variable holding the physical suitability for urban land use classes and decreased the suitability for housing because of an increased flood risk in a certain cell.

Water levels are derived from the discharge values with a stage–discharge relation. The water levels are translated to a two-dimensional water surface, and cells were identified as being flooded when the difference between the water surface and the elevation map exceeds a certain flooding threshold. The flood zones are used to determine the suitability for urban areas. Flooded cells are assigned as not suitable for housing to have a clear view on the effects of the bidirectional coupling.

The calculation of the flood zones follows the approaches described, for example, by Ward *et al.* (2011) or Haasnoot *et al.* (2012). In the first stage of the model construction, we tentatively assigned a binary value to determine the suitability for housing. This implementation could be improved by adopting a more rigorous approach, assigning for example the suitability based on damage cost functions (e.g., ICPR 2001; de Kok & Grossmann 2010).

Model implementation

Land use change is modelled by wrapping the RuimteModel into a component model of the modelling framework. The hydrological component models are implemented in Python and therefore integrate seamlessly into the modelling framework. More examples of component models can be found in Schmitz *et al.* (2013).

Figure 10 shows an excerpt of the model algebra script specifying the components and interactions of the integrated model for flood risk assessment. First, the individual components are instantiated with their simulation period and time steps (lines 2–4). The script shows two component models *Luc* and *Hyd* modelling land use change and rainfall–runoff, respectively, and a component *Demog* providing population data. All components are then added to an instance of the integrated model (lines 7–9). Next, the modeller specifies the exchange of variables between the component models. A direct transfer of variables without temporal aggregation is shown in line 12 where the population output variable of the demographic data provider is coupled to the corresponding input variable of the land use change component. An accumulator

```
1 # Creating instances of the domain models
2 luc = Luc(datetime(2010,1,1), datetime(2030,1,1), ↩
      timedelta(days=365))
3 hyd = Hyd(datetime(2010,1,1), datetime(2030,1,1), ↩
      timedelta(days=1))
4 demog = Demog(datetime(2010,1,1), datetime(2030,1,1), ↩
      timedelta(days=365))
5
6 # Add domain models to instance of integrated model
7 caseStudy += luc
8 caseStudy += hyd
9 caseStudy += demog
10
11 # Specify interactions
12 caseStudy.link(demog["population"], luc["population"])
13 adapt = ReclassResample(luc["landuse"], hyd["landuse"])
14 caseStudy += adapt
15 caseStudy.run()
```

Figure 10 | Python script specifying components and interactions of the integrated model for flood risk assessment. Not all instantiations of component models and interactions are shown.

ReclassResample needs to interconnect the two component models because the characteristics of the outgoing variables are not consistent with the characteristics of the incoming variables (line 13).

Integration of external functionality into a modelling framework

Several use cases can require the embedding of an external application into a modelling framework. The embedding is advantageous if the domain to be integrated is outside of the profession of the modeller, and required if the source code of an application cannot be modified. In addition, a time-consuming reimplementation and testing of a new model code can be avoided by integrating a previously applied model application. We use the wrapper approach to integrate the existing land use model, thereby obtaining a reusable land use change component complying with the modelling framework. Wrapping allows the component model to be used in other case studies, for example to investigate the influence of land use change on the economic quantification of ecosystem services (e.g., Kragt *et al.* 2011; Broekx *et al.* 2013).

A component model therefore acts as a proxy for the embedded model application (Hahn *et al.* 2009). However, a model application needs to fulfil certain requirements to be embedded into a modelling framework. First of all, the application must be able to proceed a number of time steps stipulated by the modelling framework. Second, all required inputs and outputs of the external application must be accessible for a modelling framework, for example by file exchange via hard disk or through a command line interface. Finally, the application must not require user interaction such that the framework can execute the application automatically.

The land use model used in this case study complies with these requirements, and its land use change processes are mapped to the component model *LucModel*. The Python script implemented by the modeller shows the wrapping of the RuimteModel (Figure 11). The *init* section specifies the temporal characteristics such as start and end time as well as the time step of the component (line 3). In addition, the state variables such as the land use map are initialised, and the input and output variables and therefore the component interface are specified. The land use model requires a specific folder structure, such as folders for required libraries, input and output files. Lines 9–10 construct path locations referring to these static elements such as the executable model, or the scenario input files.

```
1 class LucModel(PCRasterRealTimeComponent.PCRasterRealTimeComponent):
2   def __init__(self, start, end, deltaT, cloneMap):
3     PCRasterRealTimeComponent.PCRasterRealTimeComponent.__init__(self, cloneMap, start, end, deltaT)
4     self.landuse = self.readmap(cloneMap)
5     self.suitability = self.readmap(cloneMap)
6     self.addInputValue(self.suitability)
7     self.addOutputValue(self.landuse)
8     # specify fixed path settings
9     self.workingDir = os.getcwd()
10    self.exePath = os.path.join(self.workingDir, "bin", "ruimtemodel.exe")
11
12  def runTimestep(self):
13    # write new suitability map to input directory of land use change model
14    self._writeSuitability(self.suitability)
15    # generate command line string
16    currentYear = self.timeStepAsRealTime().year
17    landuseInputPath = os.path.join(self.workingDir, "kleine_nete", "lu_%d.asc" % (currentYear))
18    landuseOutputPath = os.path.join(self.workingDir, "kleine_nete", "lu_%d.asc" % (currentYear + 1))
19    cmd = "%s %s %d %s %s %f %d %s" % (self.exePath, self.simFilePath, currentYear, landuseInputPath, ruleFilePath, ↩
          self.alpha, currentYear + 1, landuseOutputPath)
20    self._executeCommand(cmd)
21    # transfer land use change model results to process component output variable
22    cmd = "gdal_translate -of PCRaster -ot Int32 -mo PCRASTER_VALUESCALE=VS_NOMINAL -a_nodata 37 %s %s" % ↩
          (self.rstPath, self.pcrPath)
23    self._executeCommand(cmd)
24    self.landuse = self.readmap(self.pcrPath)
```

Figure 11 | Python implementation of the component that models land use change. The external RuimteModel is executed by a system call. The initialisation of some variables is omitted.

The *runTimestep* section is executed for each time step. First, the land use change model is initialised with the updated inputs of the *LucModel* component. Therefore, the incoming spatial suitability map is written to the input folder of the RuimteModel (line 14). Second, the external application needs to be executed for a 1-year time step. Lines 16–20 describe the construction of the command line string carrying out the execution of the land use model. The input file location of the current land use for time step t and the output file location for time step $t+1$ are constructed. The command line string is composed of the executable and its input arguments, specifying input and output file locations and the simulation period. Finally, the new land uses calculated by the land use model need to be provided as output variables of the *LucModel* component. Lines 22–24 show this conversion using the GDAL library (2013).

Implementing accumulators with spatio-temporal scaling operations

Since the land use model and the hydrological component models use different spatial and temporal discretisations, a direct exchange of maps between these domains is not feasible.

The land use model uses 37 land use classes, a 100 m grid cell size and yearly time steps. The hydrological components use 17 land use classes, a 50 m grid cell size and daily time steps. Accumulators are therefore required in both directions for an appropriate coupling of the two domains.

Figure 12 shows the accumulator *ReclassResample* adjusting the land use output map for each year to an adequate input map for the hydrological model (see Figure 9). The accumulator is initalised with the references to the providing and receiving components (see Figure 10) including the exchanged variable, in this case the land use classes (line 3). The temporal operation g (Equation (2)) is not implemented in the accumulator (*pass* in line 6). The update method only performs operations on the spatial discretisation. The spatial scaling (f in Equation (1)) of the land use variable is implemented with operations from the PCRaster Python module (Karssenberg *et al.* 2007). The 17 new land use classes are assigned by the *lookupnominal*

```
1 class ReclassResample(Accumulator):
2  def __init__(self, inComponent, outComponent):
3    Accumulator.__init__(inComponent, outComponent)
4
5  def time_aggregate(self):
6    pass
7
8  def update(self):
9    variable = self.interval()
10   landuse = lookupnominal("luConversion.tbl", variable)
11   landuse = rescale(variable, 0.5)
12   return landuse
```

Figure 12 | Python script showing the accumulator with operations converting the number of land use classes and operations rescaling the grid cell size of the land use variable.

operation based on the conversion table given in the file *lucConversion* (line 10). The cell size is modified by rescale with a scaling factor of 0.5, directly assigning the land use classes to the 50 m grid used by WetSpa (line 11). The *update* operation returns a land use map matching the discretisations of the hydrological components.

The transfer of the output maps from the hydrological components to the land use change component model requires an aggregation of the daily discharge values to a yearly value in addition to the spatial scaling. Figure 13 shows the implementation of the corresponding accumulator. The calculation of flood zones for each year is implemented in the *time_aggregate* method (g in Equation (2)). For each day of the expired year, the discharge value is converted to a water level and subsequently to a water surface map (line 8). The difference with the elevation map

```
1 class FloodingSusceptibility(Accumulator):
2 def __init__(self, componentIn, componentOut, variable):
3  Accumulator.__init__(componentIn, componentOut, ↩
       variable)
4
5 def time_aggregate(self, variable):
6  floodTotal = boolean(0)
7  for discharge in interval():
8   waterSurface = self.getWaterSurface(discharge)
9   depth = ifthen(waterSurface > dem, waterSurface - dem)
10  flood = clump(ifthen(defined(depth), 1))
11  floodTotal = floodTotal | flood
12 return self.scale(floodTotal)
13
14 def update(self):
15 return self.time_aggregate()
```

Figure 13 | Python implementation of the accumulator interconnecting the hydrological components and the land use change component. The symbol | represents the spatial Boolean *OR*.

determines the inundation depth for each cell (line 9). The *clump* operation groups all affected cells into a flood zone. The result is a map holding Boolean *True* values in cells that were inundated in the current time step. The Boolean *OR* operation (line 11) combines the flooded cells of the current time step with the yearly total flood zones. The result is scaled to a 100 m discretisation before it is transferred to the land use change component.

Figure 14 shows output maps from the component models and accumulators. Note that these outputs are preliminary results obtained with a technically verified but uncalibrated model. The results are only included to demonstrate possible outputs of the integrated model. Figure 14(a) shows the inundation depths occurring at 1 day. Figure 14(b) shows a suitability map as a result of the aggregated flood zones of the last year. Cells affected by flooding are marked as not suitable for urban development. Figure 14(c) shows the differences in the newly allocated urban land use classes for a 10-year simulation. When simulating using a unidirectional coupling, in which only the land use in the hydrological component model is updated, only a few cells in the southwest of the study area are transformed to urban area. Incorporating a bidirectional exchange between the land use and the hydrological component models leads to a disappearance of cells classified as urban in the area susceptible to flooding.

DISCUSSION AND CONCLUSION

Component-based development practices applied to environmental modelling can lead to a more straightforward construction of generic and reusable component models. Coupling multi-scale component models and runtime interactions at model execution, however, demand the modeller to adjust exchanged variables for a sound setup of integrated models. The presented accumulator is a generic model building block suitable to hold these

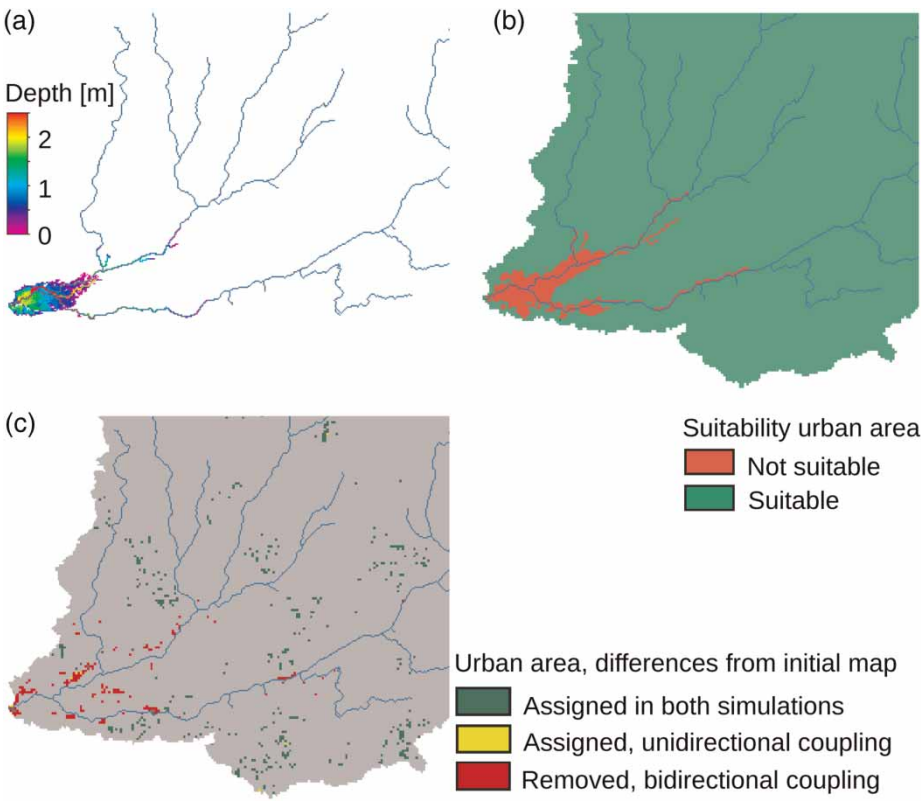

Figure 14 | Results of the integrated model at different time steps. (a) Inundation depths, derived from a daily discharge value. (b) Urban suitability, derived from accumulated flooding zones over 1 year. (c) Allocation of urban areas after two 10-year simulations with different interactions between land use change and hydrology.

adjustments. The accumulator provides a tool to implement aggregation or disaggregation operations on spatial and temporal discretisations, which can be completed by the model builder using map algebra operations. The usage of the accumulator was illustrated in a case study by constructing an integrated model for flood risk assessment consisting of several component models running at different spatial and temporal discretisations.

The component-based development practices and modular development approaches help to limit the complexity of large systems by representing natural processes with confined process representations. The design and development can be carried out without affecting other component models, a facet becoming more important in maintaining a longer lifecycle of component models (e.g., Scholten *et al.* 2007; Hahn *et al.* 2009). The need to implement spatial and temporal scaling operations in the construction of multidisciplinary models will increase when component-based model development practices are used. However, approaches that provide generic model building blocks for scaling operations that are programmable by modellers just like component models are rare. Modellers using system programming languages for model development or scripting languages as glue for component model integration (e.g., Roberts *et al.* 2010; Huang *et al.* 2012) receive no built-in support for time management, distribution of exchanged variables and implementation of scaling operations themselves. Frameworks with emphasis on the component coupling such as TDT (Hinkel 2009) or OpenMI 1.4 (e.g., Gregersen *et al.* 2007) focus on the specification on the component interfaces and can require a modification of the component interfaces or require a requested component to perform conversions (Moore & Tindall 2005). Additional dependencies can therefore be imposed on individual component models, making them less reusable in other model applications. Hence, the latest version of the OpenMI standard (e.g., Donchyts *et al.* 2010) introduces the IAdaptedOutput interface suitable for spatial scaling or unit conversions to overcome this requirement. Building blocks for numerical integration or extrapolation that are provided in graphical modelling systems such as ExtendSim (2013) lack support for spatial explicit operations. Environmental modelling frameworks such as the ESMF (e.g., Hill *et al.* 2004) or CSDMS (e.g., Peckham *et al.* 2013) allow modellers to implement scaling operations but rather on the level of programming languages than on domain concepts. The modelling framework presented in this study is an important addition to these existing frameworks, particularly because it provides a single research environment containing building blocks to implement component models, couplings and scaling operations. Calibration and data assimilation techniques can be added to the same framework comparable to Karssenberg *et al.* (2010). The result is a single software environment that supports almost all steps in integrated model construction. Modellers can therefore stay within the same software environment during model construction, which minimises the time required to invest in learning how to use the tools (Killcoyne & Boyle 2009).

A modeller can use the introduced accumulators to implement spatial and temporal aggregation or disaggregation operations, allowing for a runtime coupling of component models associated with their characteristic scales. The accumulators offer the modeller only a technical means to straightforwardly access variables required for implementing the scaling operations of coupled component models. Model builders, however, still need to regard characteristics of component models and the designated scaling operations while establishing a coupling. Obtaining knowledge about the component models is required as those may have internal space–time constraints that restrict couplings and scaling operations. The influence of parameters – obtained by calibrating the component models – on the integrated model needs to be evaluated as well. The MOLAND-based land use model and WetSpa were used as individual models in several scientific studies (e.g., White *et al.* 1997; Liu & De Smedt 2005; Engelen *et al.* 2007; Hurkens *et al.* 2008; Poelmans *et al.* 2010; Safari *et al.* 2012) but an appropriate assessment of the parameter values on the coupled model still needs to be carried out. In addition, the methodology used in an accumulator strongly influences the outcomes of the integrated models. Scaling operations might introduce errors. Aggregation, for example, inevitably causes the loss of information (e.g., van Beurden & Douven 1999). In addition, the order of operations on space and time given by the modeller will produce different results in non-linear scaling schemes. Mass balance properties need to be maintained during disaggregation, for

instance when disaggregating monthly precipitation values into daily values. Moreover, conceptual problems may arise in the coupling of component models from domains with a strong spatial modelling background such as hydrology to domains without a traditional explicit treatment of space such as economics. Finally, the introduced accumulators do not yet provide a set of standard operations calculating statistics over time on raster-based maps, for example, operations such as time average (Karssenberg & de Jong 2005b). However, this limitation is a result of the prototype implementation status of the modelling framework rather than a conceptual limitation.

The framework implementation requires a component model to store the complete history of state variables to the hard disk. Therefore, arbitrary data requests can be fulfilled from any other component. Storing the state variables of all component models for each time step, however, can require significant amounts of disk space when large spatial datasets are involved. Potential storage reduction could be explored by in-memory storage of state variables, similar to the SmartBuffer in OpenMI (see e.g., Elag *et al.* 2011).

So far, only a few studies quantify the effects involved in the coupling of multi-scale integrated models (e.g., Claessens *et al.* 2009; Moreira *et al.* 2009; Elag *et al.* 2011). Further research is needed to assess the influence of the time step of individual component models, and the choice of aggregation and disaggregation over space and time on feedback effects. We believe that the programmable accumulator provided with our modelling framework can help a modeller in the assessment of alternative couplings of integrated models.

ACKNOWLEDGEMENTS

This work was supported by research funding from the Flemish Institute for Technological Research (VITO) and the Research Foundation Flanders (FWO). We thank Guy Engelen, Jean-Luc de Kok, Jan Bronders (VITO), Steven M. de Jong, Kor de Jong, Rens van Beek and Marjolijn Haasnoot (Utrecht University) for their input at various stages of the work. We also thank the three reviewers for their constructive suggestions.

REFERENCES

Abbott, M. B., Bathurst, J. C., Cunge, J. A., O'Connell, P. E. & Rasmussen, J. 1986a An introduction to the European Hydrological System – Système Hydrologique Européen, "SHE", 1: history and philosophy of a physically-based, distributed modelling system. *J. Hydrol.* **87** (1–2), 45–59.

Abbott, M. B., Bathurst, J. C., Cunge, J. A., O'Connell, P. E. & Rasmussen, J. 1986b An introduction to the European Hydrological System – Système Hydrologique Européen, "SHE", 2: structure of a physically-based, distributed modelling system. *J. Hydrol.* **87** (1–2), 61–77.

Ahrends, H., Mast, M., Rodgers, C. & Kunstmann, H. 2008 Coupled hydrological-economic modelling for optimised irrigated cultivation in a semi-arid catchment of West Africa. *Environ. Modell. Softw.* **23** (4), 385–395.

Argent, R. M. 2004 An overview of model integration for environmental applications–components, frameworks and semantics. *Environ. Modell. Softw.* **19** (3), 219–234.

Argent, R. M. 2005 A case study of environmental modelling and simulation using transplantable components. *Environ. Modell. Softw.* **20** (12), 1514–1523.

Argent, R. M., Voinov, A., Maxwell, T., Cuddy, S. M., Rahman, J. M., Seaton, S., Vertessy, R. A. & Braddock, R. D. 2006 Comparing modelling frameworks – A workshop approach. *Environ. Modell. Softw.* **21** (7), 895–910.

Argent, R. M., Perraud, J.-M., Rahman, J. M., Grayson, R. B. & Podger, G. M. 2009 A new approach to water quality modelling and environmental decision support systems. *Environ. Modell. Softw.* **24** (7), 809–818.

Armstrong, C. W., Ford, R. W. & Riley, G. D. 2009 Coupling integrated Earth System Model components with BFG2. *Concurr. Comput. – Pract. Exp.* **21** (6), 767–791.

Barnes, B., Mokany, K. & Roderick, M. 2007 Allocation within a generic scaling framework. *Ecol. Modell.* **201** (2), 223–232.

Barredo, J. I., Kasanko, M., McCormick, N. & Lavalle, C. 2003 Modelling dynamic spatial processes: simulation of urban future scenarios through cellular automata. *Landscape Urban. Plan.* **64** (3), 145–160.

Barredo, J. I., Demicheli, L., Lavalle, C., Kasanko, M. & McCormick, N. 2004 Modelling future urban scenarios in developing countries: an application case study in Lagos, Nigeria. *Environ. Plann. B* **31** (1), 65–84.

Batelaan, O., Wang, Z.-M. & De Smedt, F. 1996 An adaptive GIS toolbox for hydrological modelling. In: *Application of Geographic Information Systems in Hydrology and Water Resources Management* (K. Kovar & H.-P. Nachtnebel, eds). Vol. 235, IAHS Press, Wallingford, UK, pp. 3–9.

Beck, K. & Andres, C. 2004 *Extreme Programming Explained: Embracing Change*, 2nd edn. Addison-Wesley, Boston, MA.

Beck, M. B., Jakeman, A. J. & McAleer, M. J. 1993 Construction and evaluation of models of environmental systems. In: *Modelling Change in Environmental Systems* (M. B. Beck,

A. J. Jakeman & M. J. McAleer, eds). John Wiley & Sons, New York, pp. 3–35.

Becker, B. P. & Schüttrumpf, H. 2011 An OpenMI module for the groundwater flow simulation programme Feflow. *J. Hydroinform.* **13** (1), 1–12.

Bierkens, M. F. P., Finke, P. A. & de Willigen, P. 2000 *Upscaling and Downscaling Methods for Environmental Research*. Kluwer, Dordrecht, The Netherlands.

Blöschl, G. & Sivapalan, M. 1995 Scale issues in hydrological modelling: a review. *Hydrol. Process.* **9** (3–4), 251–290.

Boehm, B. W. 1988 A spiral model of software development and enhancement. *Computer* **21** (5), 61–72.

Booch, G., Rumbaugh, J. & Jacobson, I. 2005 *Unified Modeling Language User Guide*, 2nd edn. Addison-Wesley, Boston, MA.

Broekx, S., Liekens, I., Peelaerts, W., De Nocker, L., Landuyt, D., Staes, J., Meire, P., Schaafsma, M., Van Reeth, W., Van den Kerckhove, O. & Cerulus, T. 2013 A web application to support the quantification and valuation of ecosystem services. *Environ. Impact Asses. Rev.* **40**, 65–74.

Bulatewicz, T., Allen, A., Peterson, J. M., Staggenborg, S., Welch, S. M. & Steward, D. R. 2013 The Simple Script Wrapper for OpenMI: enabling interdisciplinary modeling studies. *Environ. Modell. Softw.* **39**, 283–294.

Burrough, P. A. 1998 Dynamic modelling and geocomputation. In: *Geocomputation: A Primer* (P. A. Longley, S. M. Brooks, R. McDonnel & B. MacMillan, eds). Wiley, Chichester, UK, pp. 165–191.

Castronova, A. M. & Goodall, J. L. 2010 A generic approach for developing process-level hydrologic modeling components. *Environ. Modell. Softw.* **25** (7), 819–825.

Chow, V. T., Maidment, D. R. & Mays, L. W. 1988 *Applied Hydrology*. McGraw-Hill, New York.

Claessens, L., Schoorl, J. M., Verburg, P. H., Geraedts, L. & Veldkamp, A. 2009 Modelling interactions and feedback mechanisms between land use change and landscape processes. *Agr. Ecosyst. Environ.* **129** (1–3), 157–170.

Cohen, D., Lindvall, M. & Costa, P. 2004 An introduction to agile methods. *Adv. Comput.* **62** (C), 1–66.

Dams, J., Salvadore, E., Daele, T. V., Ntegeka, V., Willems, P. & Batelaan, O. 2012 Spatio-temporal impact of climate change on the groundwater system. *Hydrol. Earth Syst. Sci.* **16** (5), 1517–1531.

Dams, J., Dujardin, J., Reggers, R., Bashir, I., Canters, F. & Batelaan, O. 2013 Mapping impervious surface change from remote sensing for hydrological modeling. *J. Hydrol.* **485**, 84–95.

David, O., Ascough II, J. C., Lloyd, W., Green, T. R., Rojas, K. W., Leavesley, G. H. & Ahuja, L. R. 2013 A software engineering perspective on environmental modeling framework design: the Object Modeling System. *Environ. Modell. Softw.* **39**, 201–213.

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. Modell.* **220** (24), 3527–3535.

de Kok, J.-L. & Grossmann, M. 2010 Large-scale assessment of flood risk and the effects of mitigation measures along the Elbe River. *Nat. Hazards* **52** (1), 143–166.

de Kok, J.-L., Kofalk, S., Berlekamp, J., Hahn, B. & Wind, H. 2009 From design to application of a decision-support system for integrated river-basin management. *Water Resour. Manag.* **23** (9), 1781–1811.

de Kok, J.-L., Poelmans, L., Engelen, G., Uljee, I. & van Esch, L. 2012 Spatial-dynamic visualization of long-term scenarios for demographic, social-economic and environmental change in Flanders. In: *International Congress on Environmental Modelling and Software* (R. Seppelt, A. A. Voinov, S. Lange & D. Bankamp, eds). Managing Resources of a Limited Planet: Pathways and Visions under Uncertainty, Sixth Biennial Meeting. International Environmental Modelling and Software Society (iEMSs), Leipzig, Germany, pp. 1984–1991.

de Roo, A., Thielen, J., Salamon, P., Bogner, K., Nobert, S., Cloke, H., Demeritt, D., Younis, J., Kalas, M., Bódis, K., Muraro, D. & Pappenberger, F. 2011 Quality control, validation and user feedback of the European Flood Alert System (EFAS). *Int. J. Digit. Earth* **4**(Suppl. 1), 77–90.

De Smedt, F., Liu, Y. B. & Gebremeskel, S. 2000 Hydrologic modeling on a catchment scale using GIS and remote sensed land use information. *Manag. Informat. Syst.* 295–304.

Donchyts, G. & Jagers, B. 2010 DeltaShell – An open modelling environment. In: *Proceedings of the iEMSs Fifth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2010)* (D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli & T. Filatova, eds). Vol. 2. International Environmental Modelling and Software Society, Ottawa, Canada, pp. 1050–1057.

Donchyts, G., Hummel, S., Vaneçek, S., Groos, J., Harper, A., Knapen, R., Gregersen, J., Schade, P., Antonello, A. & Gijsbers, P. 2010 OpenMI 2.0 – What's new? In: *Modelling for Environment's Sake: Proceedings of the 5th Biennial Conference of the International Environmental Modelling and Software Society, iEMSs 2010* (D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli & T. Filatova, eds). Vol. 2. International Environmental Modelling and Software Society (iEMSs), pp. 1174–1181.

Elag, M. & Goodall, J. L. 2013 An ontology for component-based models of water resource systems. *Water Resour. Res.* **49** (8), 5077–5091.

Elag, M. M., Goodall, J. L. & Castronova, A. M. 2011 Feedback loops and temporal misalignment in component-based hydrologic modeling. *Water Resour. Res.* **47** (12), W12520.

Engelen, G., Hagen-Zanker, A., de Nijs, A. C. M., Maas, A., van Loon, J., Straatman, B., White, R., Uljee, I., van der Meulen, M. & Hurkens, J. 2005 Kalibratie en validatie van de LeefOmgevings Verkenner (Calibration and validation of the LeefOmgevings Verkenner) (in Dutch). Available at: http://www.pbl.nl/publicaties/2005/KalibratieEnValidatieVanDeLeefOmgevingsVerkenner.

Engelen, G., Lavalle, C., Barredo, J. I., van der Meulen, M. & White, R. 2007 The Moland modelling framework for urban and regional land-use dynamics. In: *Modelling Land-Use Change* (E. Koomen, J. Stillwell, A. Bakema & H. J. Scholten, eds). Vol. 90, The Geo Journal Library. Springer, Dordrecht, The Netherlands, pp. 297–320.

Engelen, G., Poelmans, L., Uljee, I., De Kok, J.-L. & Van Esch, L. 2011 De Vlaamse Ruimte in 4 Wereldbeelden – Scenarioverkenning 2050 – Kwantitatieve Verwerking (The Flemish Space in 4 World Views – Scenario Exploration 2050 – Quantification). A study for the Policy Research Center for Spatial Planning and Housing. VITO Report 2011/RMA/R/363. Flemish Institute for Technological Research (VITO), Mol, Belgium (in Dutch).

ExtendSim 2013 Imagine That product website. Available at: http://www.extendsim.com/.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. 1995 *Design Patterns: Elements of Reusable Object-oriented Software*. Addison Wesley, Boston, MA.

GDAL Development Team 2013 GDAL – Geospatial Data Abstraction Library. Available at: http://www.gdal.org/.

Granell, C., Díaz, L., Schade, S., Ostländer, N. & Huerta, J. 2013a Enhancing integrated environmental modelling by designing resource-oriented interfaces. *Environ. Modell. Softw.* **39**, 229–246.

Granell, C., Schade, S. & Ostländer, N. 2013b Seeing the forest through the trees: a review of integrated environmental modelling tools. *Comput. Environ. Urban* **41**, 136–150.

Gregersen, J. B., Gijsbers, P. J. A. & Westen, S. J. P. 2007 OpenMI: open modelling interface. *J. Hydroinform.* **9** (3), 175–191.

Güntner, A., Olsson, J., Calver, A. & Gannon, B. 2001 Cascade-based disaggregation of continuous rainfall time series: the influence of climate. *Hydrol. Earth Syst. Sci.* **5** (2), 145–164.

Haasnoot, M., Middelkoop, H., Offermans, A., van Beek, E. & van Deursen, W. P. A. 2012 Exploring pathways for sustainable water management in river deltas in a changing environment. *Clim. Change* **115** (3–4), 795–819.

Hagen-Zanker, A., van Loon, J., Maas, A., Straatman, B., de Nijs, T. & Engelen, G. 2005 Measuring performance of land use models: an evaluation framework for the calibration and validation of integrated land use models featuring cellular automata. In: *14th European Colloquium on Theoretical and Quantitative Geography*, Lisbon, Portugal.

Hahn, B. M., Kofalk, S., De Kok, J.-L., Berlekamp, J. & Evers, M. 2009 Elbe DSS: a planning support system for strategic river basin planning. In: *Planning Support Systems Best Practice and New Methods* (S. Geertman & J. Stillwell, eds). Springer, pp. 113–136.

Hall, J. & Solomatine, D. 2008 A framework for uncertainty analysis in flood risk management decisions. *Int. J. River Basin Manage.* **6** (2), 85–98.

Harvey, H., Hall, J. & Peppé, R. 2012 Computational decision analysis for flood risk management in an uncertain future. *J. Hydroinform.* **14** (3), 537–561.

Hill, C., DeLuca, C., Balajic Suarez, M. & Da Silva, A. 2004 The architecture of the Earth system modeling framework. *Comput. Sci. Eng.* **6** (1), 18–28.

Hinkel, J. 2009 The PIAM approach to modular integrated assessment modelling. *Environ. Modell. Softw.* **24** (6), 739–748.

Holzworth, D. P., Huth, N. I. & de Voil, P. G. 2010 Simplifying environmental model reuse. *Environ. Modell. Softw.* **25** (2), 269–275.

Huang, J., Gao, J., Hörmann, G. & Mooij, W. M. 2012 Integrating three lake models into a Phytoplankton Prediction System for Lake Taihu (Taihu PPS) with Python. *J. Hydroinform.* **14** (2), 523–534.

Huang, M., Maidment, D. R. & Tian, Y. 2011 Using SOA and RIAs for water data discovery and retrieval. *Environ. Modell. Softw.* **26** (11), 1309–1324.

Hunter, J. D. 2007 Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9** (3), 99–104.

Hurkens, J., Hahn, B. & van Delden, H. 2008 Using the GEONAMICA software environment for integrated dynamic spatial modelling. In: *Integrating Sciences and Information Technology for Environmental Assessment and Decision Making* (M. Sànchez-Marrè, J. Béjar, J. Comas, A. Rizzoli & G. Guariso, eds). iEMSs 2008: International Congress on Environmental Modelling and Software, pp. 751–758.

ICPR 2001 *Abschlussbericht*. Vorgehensweise zur Ermittlung der hochwassergefährdeten Flächen und der möglichen Vermögenschäden (Final Report. Approach to investigate the areas at flood risk and potential economic damage). Koblenz, Germany (in German).

Jagers, H. R. A. 2010 Linking data, models and tools: an overview. In: *Proceedings of the iEMSs Fifth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2010)* (D. A. Swayne, W. Yang, A. A. Voinov, A. Rizzoli & T. Filatova, eds). International Environmental Modelling and Software Society, Ottawa, Canada, pp. 1150–1157.

Jakeman, A. J. & Letcher, R. A. 2003 Integrated assessment and modelling: features, principles and examples for catchment management. *Environ. Modell. Softw.* **18** (6), 491–501.

Janssen, S., Athanasiadis, I. N., Bezlepkina, I., Knapen, R., Li, H., Domínguez, I. P., Rizzoli, A. E. & van Ittersum, M. K. 2011 Linking models for assessing agricultural land use change. *Comput. Electron. Agr.* **76** (2), 148–160.

Joppich, W. & Kürschner, M. 2006 MpCCI-a tool for the simulation of coupled applications. *Concurr. Comput. – Pract. Exp.* **18** (2), 183–192.

Karaouzas, I., Dimitriou, E., Skoulikidis, N., Gritzalis, K. & Colombari, E. 2009 Linking hydrogeological and ecological tools for an integrated river catchment assessment. *Environ. Model. Assess.* **14** (6), 677–689.

Karssenberg, D. 2006 Upscaling of saturated conductivity for Hortonian runoff modelling. *Adv. Water Resour.* **29** (5), 735–759.

Karssenberg, D. & de Jong, K. 2005a Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. *Int. J. Geogr. Inf. Sci.* **19** (5), 559–579.

Karssenberg, D. & de Jong, K. 2005b Dynamic environmental modelling in GIS: 2. Modelling error propagation. *Int. J. Geogr. Inf. Sci.* **19** (6), 623–637.

Karssenberg, D., de Jong, K. & van der Kwast, J. 2007 Modelling landscape dynamics with Python. *Int. J. Geogr. Inf. Sci.* **21** (5), 483–495.

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

Kiehle, C., Greve, K. & Heier, C. 2007 Requirements for Next Generation Spatial Data Infrastructures-Standardized Web Based Geoprocessing and Web Service Orchestration. *Trans. GIS* **11** (6), 819–834.

Killcoyne, S. & Boyle, J. 2009 Managing chaos: lessons learned developing software in the life sciences. *Comput. Sci. Eng.* **11** (6), 20–29.

Knapen, R., Janssen, S., Roosenschoon, O., Verweij, P., de Winter, W., Uiterwijk, M. & Wien, J.-E. 2013 Evaluating OpenMI as a model integration platform across disciplines. *Environ. Modell. Softw.* **39**, 274–282.

Kniberg, H. 2011 *Lean from the Trenches: Managing Large-Scale Projects with Kanban*. Pragmatic Bookshelf, Dallas, TX.

Kragt, M. E., Newham, L. T. H., Bennett, J. & Jakeman, A. J. 2011 An integrated approach to linking economic valuation and catchment modelling. *Environ. Modell. Softw.* **26** (1), 92–102.

Langtangen, H. P. 2007 *Python Scripting for Computational Science*, 3rd Edition. Springer-Verlag, Berlin.

Liu, J., Peng, C., Dang, Q., Apps, M. & Jiang, H. 2002 A component object model strategy for reusing ecosystem models. *Comput. Electron. Agr.* **35** (1), 17–33.

Liu, Y. B. & De Smedt, F. 2005 Flood modeling for complex terrain using GIS and remote sensed information. *Water Resour. Manage.* **19** (5), 605–624.

Liu, Y. B., Gebremeskel, S., Smedt, F. D., Hoffmann, L. & Pfister, L. 2003 A diffusive transport approach for flow routing in GIS-based flood modeling. *J. Hydrol.* **283** (1–4), 91–106.

López, I. F. V., Snodgrass, R. T. & Moon, B. 2005 Spatiotemporal aggregate computation: a survey. *IEEE Trans. Knowl. Data Eng.* **17** (2), 271–286.

Malone, B. P., McBratney, A. B., Minasny, B. & Wheeler, I. 2012 A general method for downscaling earth resource information. *Comput. Geosci.* **41**, 119–125.

McIntosh, B. S., Jeffrey, P., Lemon, M. & Winder, N. 2005 On the design of computer-based models for integrated environmental science. *Environ. Manage.* **35** (6), 741–752.

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

Mineter, M. J., Jarvis, C. H. & Dowers, S. 2003 From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with interpolated climate data. *Environ. Modell. Softw.* **18** (4), 379–391.

Monninkhoff, B. & Kernbach, K. 2006 Coupled surface water – groundwater modelling for planning of flood retention in the lower havel area. In: *Proceedings of the International FEFLOW User Conference*, 10–15 September, Berlin, pp. 115–124.

Moore, R. V. & Tindall, C. I. 2005 An overview of the open modelling interface and environment (the OpenMI). *Environ. Sci. Policy* **8** (3), 279–286.

Moreira, E., Costa, S., Aguiar, A., Câmara, G. & Carneiro, T. 2009 Dynamical coupling of multiscale land change models. *Landsc. Ecol.* **24** (9), 1183–1194.

Morita, M. & Yen, B. C. 2002 Modeling of conjunctive two-dimensional surface-three-dimensional subsurface flows. *J. Hydraul. Eng.* **128** (2), 184–200.

Muetzelfeldt, R. & Massheder, J. 2003 The Simile visual modelling environment. *Eur. J. Agron.* **18** (3–4), 345–358.

OC GIS-Vlaanderen 2001 *Digital version of Land-cover Data Set of Flanders 2001*. Agentschap voor geografische Informatie Vlaanderen, Ghent, Belgium.

OpenMI 2 SDK 2013 Open Modelling Interface Software Development Kit. Available at: http://www.openmi.org/.

Oreskes, N., Shrader-Frechette, K. & Belitz, K. 1994 Verification, validation, and confirmation of numerical models in the earth sciences. *Science* **263** (5147), 641–646.

Papajorgji, P. 2005 A plug and play approach for developing environmental models. *Environ. Modell. Softw.* **20** (10), 1353–1357.

PCRaster 2013 PCRaster website. Available at: http://www.pcraster.eu.

Peckham, S. D., Hutton, E. W. H. & Norris, B. 2013 A component-based approach to integrated modeling in the geosciences: The design of CSDMS. *Comput. Geosci.* **53**, 3–12.

Poelmans, L., Van Rompaey, A. & Batelaan, O. 2010 Coupling urban expansion models and hydrological models: how important are spatial patterns? *Land Use Policy* **27** (3), 965–975.

Pullar, D. 2004 SimuMap: a computational system for spatial modelling. *Environ. Modell. Softw.* **19** (3), 235–243.

Python 2013 Python programming language website. Available at: http://www.python.org/.

Rastetter, E. B., Aber, J. D., Peters, D. P. C., Ojima, D. S. & Burke, I. C. 2003 Using mechanistic models to scale ecological processes across space and time. *BioScience* **53** (1), 68–76.

R Development Core Team 2013 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. Available at: http://www.R-project.org.

Refsgaard, J. C., van der Sluijs, J. P., Højberg, A. L. & Vanrolleghem, P. A. 2007 Uncertainty in the environmental modelling process – A framework and guidance. *Environ. Modell. Softw.* **22** (11), 1543–1556.

Rizzoli, A. E., Donatelli, M., Athanasiadis, I. N., Villa, F. & Huber, D. 2008 Semantic links in integrated modelling frameworks. *Math. Comput. Simulat.* **78** (2–3), 412–423.

Roberts, J. J., Best, B. D., Dunn, D. C., Treml, E. A. & Halpin, P. N. 2010 Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++. *Environ. Modell. Softw.* **25** (10), 1197–1207.

Rotmans, J. 1990 *IMAGE: An Integrated Model to Assess the Greenhouse Effect*. Kluwer, Dordrecht, The Netherlands.

Rubin, K. S. 2012 *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, New York.

Safari, A., De Smedt, F. & Moreda, F. 2012 WetSpa model application in the Distributed Model Intercomparison Project (DMIP2). *J. Hydrol.* **418–419**, 78–89.

Sargent, R. G. 2013 Verification and validation of simulation models. *J. Simulat.* **7** (1), 12–24.

Scheller, R. M., Sturtevant, B. R., Gustafson, E. J., Ward, B. C. & Mladenoff, D. J. 2010 Increasing the reliability of ecological models using modern software engineering techniques. *Front. Ecol. Environ.* **8** (5), 253–260.

Schmitz, O., Karssenberg, D., van Deursen, W. P. A. & Wesseling, C. G. 2009 Linking external components to a spatio-temporal modelling framework: coupling MODFLOW and PCRaster. *Environ. Modell. Softw.* **24** (9), 1088–1099.

Schmitz, O., Karssenberg, D., de Jong, K., de Kok, J.-L. & de Jong, S. M. 2013 Map algebra and model algebra for integrated model building. *Environ. Modell. Softw.* **48**, 113–128.

Schmolke, A., Thorbek, P., DeAngelis, D. L. & Grimm, V. 2010 Ecological models supporting environmental decision making: a strategy for the future. *Trends Ecol. Evol.* **25** (8), 479–486.

Scholten, H., Kassahun, A., Refsgaard, J. C., Kargas, T., Gavardinas, C. & Beulens, A. J. 2007 A methodology to support multidisciplinary model-based water management. *Environ. Modell. Softw.* **22** (5), 743–759.

Schweitzer, C., Priess, J. A. & Das, S. 2011 A generic framework for land-use modelling. *Environ. Modell. Softw.* **26** (8), 1052–1055.

Skøien, J. O., Blöschl, G. & Western, A. W. 2003 Characteristic space scales and timescales in hydrology. *Water Resour. Res.* **39** (10), SWC111–SWC1119.

Stasch, C., Foerster, T., Autermann, C. & Pebesma, E. 2012 Spatio-temporal aggregation of European air quality observations in the Sensor Web. *Comput. Geosci.* **47**, 111–118.

Steiniger, S. & Hay, G. J. 2009 Free and open source geographic information tools for landscape ecology. *Ecol. Inform.* **4** (4), 183–195.

STELLA 2013 STELLA product website. Available at: http://www.iseesystems.com/.

Szyperski, C. 2002 *Component Software: Beyond Object-Oriented Programming*. 2nd edn, ACM Press, New York.

Tomlin, C. D. 1990 *Geographic Information Systems and Cartographic Modeling*. Prentice Hall, New York.

van Beurden, A. U. C. J. & Douven, W. J. A. M. 1999 Aggregation issues of spatial information in environmental research. *Int. J. Geogr. Inf. Sci.* **13** (5), 513–527.

Van Der Walt, S., Colbert, S. C. & Varoquaux, G. 2011 The NumPy array: a structure for efficient numerical computation. *Comput. Sci. Eng.* **13** (2), 22–30.

van Deursen, W. & Maes, J. 2008 Progress on PCRaster-Extend Interface (Update Report), Deliverable 8.6, Workpackage 8, Model Support. Tech. rep., Science Policy Interface for Coastal Systems Assessment (SPICOSA), EU 6th Framework Research Project nr. 0369927.

van Ittersum, M. K., Ewert, F., Heckelei, T., Wery, J., Alkan Olsson, J., Andersen, E., Bezlepkina, I., Brouwer, F., Donatelli, M., Flichman, G., Olsson, L., Rizzoli, A. E., van der Wal, T., Wien, J. E. & Wolf, J. 2008 Integrated assessment of agricultural systems – A component-based framework for the European Union (SEAMLESS). *Agric. Syst.* **96** (1–3), 150–165.

Vermaat, J. E., Eppink, F., van den Bergh, J. C. J. M., Barendregt, A. & van Belle, J. 2005 Aggregation and the matching of scales in spatial economics and landscape ecology: empirical evidence and prospects for integration. *Ecol. Econ.* **52** (2), 229–237.

Verweij, P. J. F. M., Knapen, M. J. R., de Winter, W. P., Wien, J. J. F., te Roller, J. A., Sieber, S. & Jansen, J. M. L. 2010 An IT perspective on integrated environmental modelling: the SIAT case. *Ecol. Modell.* **221** (18), 2167–2176.

Villa, F. 2007 A semantic framework and software design to enable the transparent integration, reorganization and discovery of natural systems knowledge. *J. Intell. Inf. Syst.* **29** (1), 79–96.

Voinov, A. & Cerco, C. 2010 Model integration and the role of data. *Environ. Modell. Softw.* **25** (8), 965–969.

Voinov, A., Fitz, C., Boumans, R. & Costanza, R. 2004 Modular ecosystem modeling. *Environ. Modell. Softw.* **19** (3), 285–304.

Wang, Z.-M., Batelaan, O. & De Smedt, F. 1996 A distributed model for water and energy transfer between soil, plants and atmosphere (WetSpa). *Phys. Chem. Earth* **21** (3 Special issue), 189–193.

Ward, P. J., De Moel, H. & Aerts, J. C. J. H. 2011 How are flood risk estimates affected by the choice of return-periods? *Nat. Hazards Earth Syst. Sci.* **11** (12), 3181–3195.

Warner, J. C., Perlin, N. & Skyllingstad, E. D. 2008 Using the Model Coupling Toolkit to couple earth system models. *Environ. Modell. Softw.* **23** (10–11), 1240–1249.

Werner, M., Schellekens, J., Gijsbers, P., van Dijk, M., van den Akker, O. & Heynert, K. 2013 The Delft-FEWS flow forecasting system. *Environ. Modell. Softw.* **40**, 65–77.

Wesseling, C. G., Karssenberg, D., Burrough, P. A. & van Deursen, W. P. A. 1996 Integrating dynamic environmental models in GIS: the development of a Dynamic Modelling language. *Trans. GIS* **1** (1), 40–48.

White, R., Engelen, G. & Uljee, I. 1997 The use of constrained cellular automata for high-resolution modelling of urban land-use dynamics. *Environ. Plann. B* **24** (3), 323–343.

First received 24 June 2013; accepted in revised form 2 October 2013. Available online 6 December 2013