HYDROLOGICAL PROCESSES
Hydrol. Process. **16**, 2751–2766 (2002)
Published online 23 August 2002 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/hyp.1068

The value of environmental modelling languages for building distributed hydrological models

Derek Karssenberg*
The Netherlands Centre for Geo-ecological Research (ICG), Faculty of Geographical Sciences, Utrecht University, PO Box 80115, 3508 TC Utrecht, The Netherlands

Abstract:

An evaluation is made of the suitability of programming languages for hydrological modellers to create distributed, process-based hydrological models. Both system programming languages and high-level environmental modelling languages are evaluated based on a list of requirements for the optimal programming language for such models. This is illustrated with a case study, implemented using the PCRaster environmental modelling language to create a distributed, process-based hydrological model based on the concepts of KINEROS-EUROSEM. The main conclusion is that system programming languages are not ideal for hydrologists who are not computer programmers because the level of thinking of these languages is too strongly related to specialized computer science. A higher level environmental modelling language is better in the sense that it operates at the conceptual level of the hydrologist. This is because it contains operators that identify hydrological processes that operate on hydrological entities, such as two-dimensional maps, three-dimensional blocks and time-series. The case study illustrates the advantages of using an environmental modelling language as compared with system programming languages in fulfilling requirements on the level of thinking applied in the language, the reusability of the program code, the lack of technical details in the program, a short model development time and learnability. The study shows that environmental modelling languages are equally good as system programming languages in minimizing programming errors, but are worse in generic application and performance. It is expected that environmental modelling languages will be used in future mainly for development of new models that can be tailored to modelling aims and the field data available. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS distributed hydrological modelling; programming languages; geographical information systems; runoff modelling

INTRODUCTION

Since the 1980s several major hydrological research groups have been developing distributed process-based hydrological models for simulating the transport of water, soil, nutrients and pollutants. Examples are groundwater transport models (e.g. Zheng, 1990; Harbaugh and McDonald, 1996; AQUA3D, 2001), rainfall-runoff models (e.g. SHE, Abbott, 1986a,b; TOPMODEL, Beven, 1997; LISFLOOD, De Roo *et al.*, 2000), rainfall-runoff models including erosion (e.g. Grayson *et al.*, 1992; EUROSEM, Morgan *et al.*, 1998; Tucker *et al.*, 1999), and rainfall-runoff models with nutrient or pollutant transport (e.g. Mackay and Ban, 1997). A single internet search on hydrological + modelling delivers hundreds of responses. Clearly, hydrologists have a continuing need for new and better models, because concepts on how to represent hydrological processes in computer simulation models are still evolving. This change of ideas in modelling is being driven by new observation techniques, including remote sensing, and data storage and presentation technology such as geographical information systems (GIS), which provide larger volumes of useful data than ever before. As with other areas of science such as astronomy or biology, new methods of data collection and processing may improve scientific understanding in ways that were not possible before they were introduced.

* Correspondence to: Derek Karssenberg, The Netherlands Centre for Geo-ecological Research (ICG), Faculty of Geographical Sciences, Utrecht University, PO Box 80115, 3508 TC Utrecht, The Netherlands. E-mail: d.karssenberg@geog.uu.nl.

Received 8 January 2001
Accepted 26 October 2001

The development of new numerical models has always been restricted by the functionality of programming languages and computer power. This will continue to be so in the future, as model demands on computers and programming languages increase, owing to a further refinement of model concepts, larger data sets and a wider application of calculation intensive methods such as Monte Carlo simulation. Although computers have opened up new research fields in hydrology, at the same time they pose restrictions, and the history of hydrological modelling has been influenced by the capacity of computers and the languages to program them.

In the twentieth century we saw a gradual transition from stand-alone programs for hydrological modelling, developed by small research groups who were at the same time developers and users, towards off-the-shelf computer programs with a user friendly interface, linked to GIS, for a generic wide application. Initially, there was little unity in hydrological modelling and every research group wrote their own software, in system programming languages such as C++ or FORTRAN.

As time went by, the number of models being worked on reduced as hydrologists selected a small set of models that embodied good science and straightforward implementation. These are the models that have been linked to GIS. As noted above, GIS can be used to supply much information for hydrological modelling, ranging from digital elevation models (DEM) of the land surface to time-series of groundwater levels and river flows. Hydrologists started to link these GIS databases to their models so that the input and output aspects of hydrological modelling could be simplified and visualized, and the results placed in a spatial and temporal context. Standard models were coupled to GIS following the loose coupling or tight coupling approach (Burrough, 1996). Loose coupling involved *ad hoc* manual exchange of data between a model with a proprietary GIS. As manual data exchange is susceptible to errors, software for automatic data exchange was written and some GIS firms began to hard-wire the hydrological models into their systems, which was called tight coupling. This provided hydrologists with a ready-to-use modelling tool, which was much favoured by consultants but not by scientists. The reason for the disfavour by scientists is that the process-understanding and algorithms are rarely state-of-the-art and also that the program code is usually inaccessible or difficult to change in such systems.

As these standard hard-wired models are of limited value for scientific research, hydrologists wishing to develop new models are left with two options. The first approach is to develop new models from scratch, or to use blocks of code from others, via a system programming language, and to link it to an existing GIS. The second, more recent approach, is to use an environmental modelling language (EML) running inside a GIS, which is known as embedded coupling (Burrough, 1996). Unlike system programming languages, which are generic purpose languages, EML are higher level programming languages with a specific application domain, in our case hydrological model construction. The approaches adopted by EML are along the following lines (Wesseling *et al*., 1996a; Karssenberg *et al*., 2001).

1. Provide a set of operators operating on spatio-temporal data in which widely accepted generic hydrological processes have been coded using accepted, clearly understood algorithms.
2. Provide these operators in a suitable way that they can be glued together in a model by a hydrologist using his or her hydrological understanding, rather than computer expertise.
3. Embed this set of tools for model construction in a GIS-like software environment providing database management and generic visualization routines for the spatio-temporal data read and written by the model.
4. Provide standard interfaces to other programming languages so that new or alternative operators can be added by the user in ways that are fully compatible with the EML.

The range of responses to the challenge of developing EML has been large, although most of them do not fulfil all four concepts given above. Some hydrologists (e.g. Olsthoorn, 1998) have used spreadsheet programs for modelling, a step that Campbell (1985) termed a 'revolution in groundwater modelling', or technical computing languages such as MATLAB (MATLAB, 2001). Although very powerful, such languages lack an embedded coupling with a GIS. Others developed graphical modelling languages with an easy to use interface for model construction (ModelMaker, 2001; STELLA, 2001). These are very powerful for process modelling,

but their non-spatial operators do not provide sufficient functionality for hydrological modelling. Modelling languages included in many GIS have the advantage that they come with powerful database and visualization tools and that they are, per definition, spatial. On the other hand, the modelling languages of proprietary GIS (e.g. ESRI, 2001) are too much focused on database management and static operations, to fulfil the requirements for spatio-temporal hydrological modelling. The most interesting developments, however, have been those made by specialist groups, who have created languages along the four concepts of EML given above. These include products such as GRASS (GRASS, 2001), PCRaster (PCRaster, 2001; Van Deursen, 1995) and Simile (Simile, 2001). The number of hydrologists using these EML is small compared with those using system programming, partly because of their more recent development. However, EMLs have proven their usefulness in model building, and it is time for an objective evaluation of this addition to the hydrologists' tool box.

Therefore, this paper will evaluate the concepts and application of EML for hydrological model building. The outline is as follows: first, a set of requirements of a programming language for hydrological model construction is defined. Second, a case study illustrates the creation of a runoff model similar to the hydrological component of KINEROS-EUROSEM (Morgan *et al*., 1998), and demonstrates how EMLs differ from system programming languages. The construction of the case study model in EML is described using PCRaster. By evaluating this software implementation of the runoff model, EMLs are compared with system programming languages on the basis of the requirements of a programming language for hydrological model construction given in the first part of the paper.

A PROGRAMMING LANGUAGE FOR HYDROLOGICAL MODEL BUILDING: REQUIREMENTS

There is a continuous need for new and better models, because concepts on how to represent hydrological processes in computer simulation models are still evolving. Models evolve partly because of changing ideas in hydrology, but also because of the expanding availability of input data, and the increasing capability to handle them using GIS, as noted in the introduction. As field data are important inputs to models, but also crucial for model calibration and validation, it is necessary that the process descriptions in the model are tailored to the data available. Models ignoring important processes that can be fed with input data will be too simple, whereas models representing processes for which no data are available will be unnecessarily complex (Van der Perk, 1997). In addition, the model process descriptions need to be tailored to the aims of modelling. Modelling the peak discharges of a river system could be done with a lumped, or simple spatial model in many cases, whereas predicting surface runoff and erosion needs a more complex spatial model. As both the data available and the aim of the model will be different for each study, it could be said that each study needs a new model, or at least a model that can be modified compared with models developed for previous studies. This does not mean that each model is unique in all its details, because there are concepts in hydrology that have proven their generic application, such as interception equations or surface water routing techniques, which are applicable in many models. The main challenge in model building is to find the optimal generic process representations and an appropriate way to combine these for a specific purpose.

If we look at model building in more detail, model development can be regarded as a process whereby different candidate model structures are evaluated until an optimal model formulation is found (Van Deursen *et al*., 2000). Most modelling studies involve such a model development cycle (Figure 1), although it is mostly not described in reports. There are, however, examples of studies describing or explicitly focusing on such a comparison between different model structures (e.g. Grayson *et al*., 1992; Van der Perk, 1997; Donelly-Makowecki and Moore, 1999). The model development cycle involves three phases per candidate model (Figure 1). In the first phase, the mathematical description of the model is defined, based on knowledge of hydrological processes and how they interact in the study area. It contains the mathematical equations simulating the set of hydrological processes. The computer program of the model, written in the second phase, is the numerical representation of this mathematical description. In the third phase, this computer

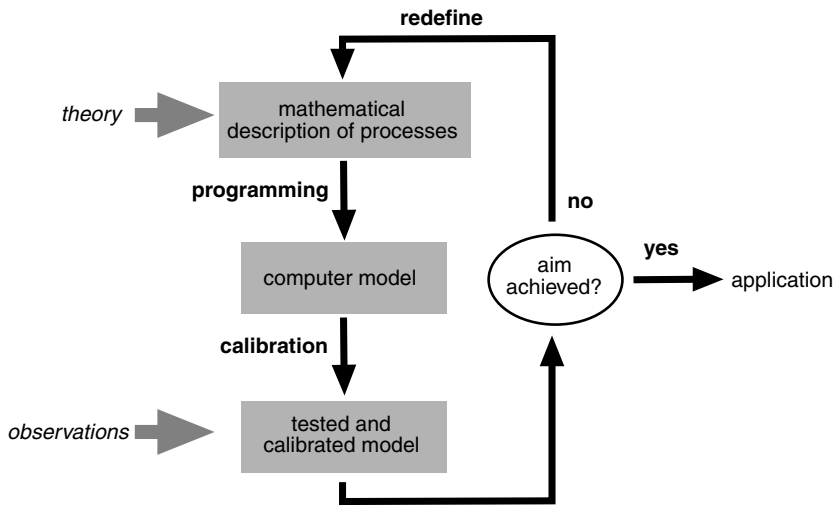

Figure 1. Model development cycle

model is tested by evaluating whether it fulfils the aim of the study or not. This involves mostly calibration of the model with field data. This phase may reveal weaknesses in the model. If weaknesses are found, attempts are made to improve the next candidate model structure: the mathematical description of the model is redefined, and a new program is written and tested against field data. Evaluating better and better candidate models is continued until a model is found that fulfils the aims of the modelling study.

The second phase in the model development cycle involves programming. It is here that the choice of the programming language becomes important. The choice of the programming language is currently not a key issue in scientific literature, probably because model implementation is regarded as a technical detail, unrelated to hydrology. If we look at Figure 1, however, we see that the language used for model implementation may have a major impact on the results of a modelling study because it affects the whole model development cycle. For instance, if it is possible with the language to change the model without too much programming, it allows evaluation of a larger number of different, candidate models, simply because it is practically more feasible. Also, the efficiency of converting the mathematical description of processes to a program code of the computer model is highly dependent on the computer language used. If the computer language is difficult to handle by hydrologists, specialist programmers are needed for software implementation, and evaluating different candidate models becomes the work of a team of programmers and hydrologists, where the hydrologist has to explain to the programmer how each candidate model should be programmed. Instead, a modelling language that can be used by the hydrologist would permit prompt software implementation of a new mathematical description of a process by the hydrologist, allowing for interactive changing of the model and evaluating its output. Other issues related to the choice of the language are also important here, such as performance of the model, or the chance of errors in the code.

For judging between different programming languages, a list of requirements for the optimal computer programming language for hydrological model development is needed. Based on a list of criteria for computer languages in Highman (1967), and with the model development cycle in mind, the following list of requirements for a language for hydrological model construction has been formulated.

1. *Level of thinking of hydrologists*. The level of thinking of a hydrologist should be represented in the model program code of models. If the concepts of the computer language represent those of hydrology, it allows easy conversion of the mathematical description of processes to program code. Also, if the program code resembles hydrological concepts, the language is more accessible to hydrologists. As a result, hydrologists

can program models themselves, without the need for specialist programmers. In addition, a language operating at the level of thinking of hydrologists enables easy exchange of program code between researchers because programs can be read by hydrologists.

2. *Reuse of program code*. Different hydrological models use similar standard operations and algorithms for the simulation of processes. For each process there are many programs that perform the same computations according to identical equations and algorithms described in the literature. For constructing a new model that needs a different combination of processes it should be possible to reuse and combine the program code of existing models. A programming language is needed that allows reuse of blocks of code already written for simulating specific processes (e.g. Harbaugh and McDonald, 1996; GMS, 1998; Leavesley *et al*., 1998). For model development, it should be easy to link these blocks together without the burden of complex programming.
3. *Generic approach to common problems*. It should be possible to make any type of distributed process-based model with the language, including (sub-) models for simulating environmental processes related to process-based hydrological modelling such as plant growth, received solar radiation and land degradation.
4. *No technical computer details*. Most developers of hydrological models are hydrologists, not computer programmers. So a programming language for model development should relieve the researcher from technical computer details that would distract her or him from scientific research.
5. *Short development time*. A programming language resulting in shorter development times would allow modifications to existing models or construction of completely new models within the framework of one modelling study, thereby tailoring the model to the modelling aims and available data. From the viewpoint of the model development cycle (Figure 1), a short development time is even more important. If it were possible to construct a new candidate model by changing an existing candidate model without too much programming work, it would allow evaluation of more different candidate models.
6. *Minimizing programming errors*. The possibility of making programming errors in hydrological model development should be as low as possible. Errors easily occur because hydrological models are large and complex; having many different processes being simulated by complicated numerical solution schemes. It should be easy to detect these errors.
7. *High performance*. As distributed hydrological models use large data sets and computationally intensive algorithms, execution times can be a problem and a programming language is needed that minimizes computation time. This requirement does not have the highest priority because the performance of computers is still doubling every 2 years.
8. *Easy to learn for hydrologists*. The programming language should be easy to learn for hydrologists who do not necessarily have programming experience.

MATHEMATICAL DEFINITION OF THE RUNOFF MODEL

The case study used in this paper for the evaluation of different programming languages is a runoff model aimed at simulating the effect of different patterns and directions of runoff pathways in agricultural catchments on the shape of the hydrograph at different locations. The model should describe the processes of interception, surface storage, infiltration and runoff with process-based equations, as detailed, spatially and temporally distributed, field data are available, including maps of preferential runoff pathways over fields. Most of these processes are included in KINEROS-EUROSEM (Morgan *et al*., 1998) but this model does not simulate preferential runoff directions on fields caused by agricultural operations. So a new model has been constructed that includes data on the pattern of runoff pathways over fields. All other process descriptions are taken from EUROSEM, schematized in Figure 2.

For each time-step, the net rainfall (P_n, m per time-step) reaching the ground is

$$P_n = P - c \cdot C \tag{1}$$

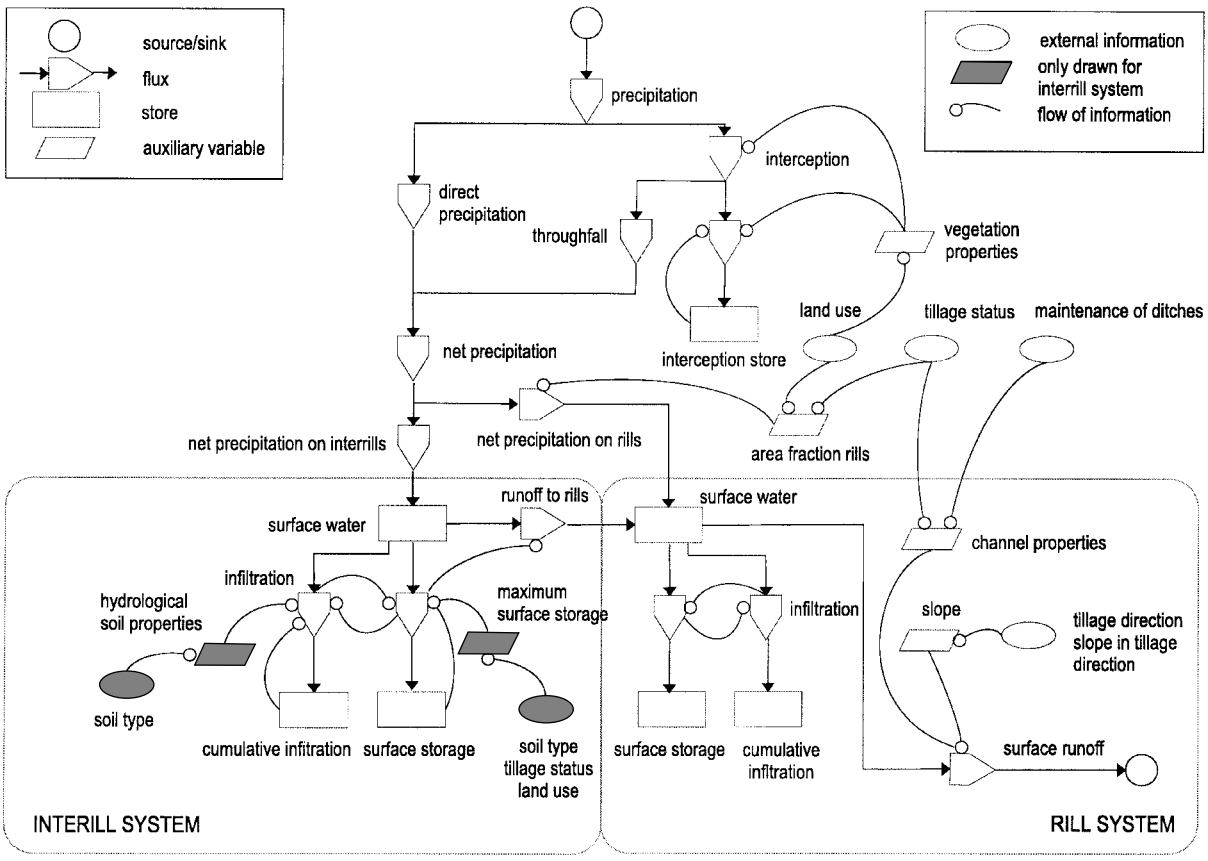

Figure 2. Flow diagram of the runoff model

where P is the open field rainfall in metres per time-step, c is the percentage cover of the vegetation (m^2/m^2) and C is the amount of water transported to the interception store, in metres per time-step for the area covered with vegetation. The value of C is (Merriam, 1973) obtained by

$$C = S_m \left(e^{\left(\frac{-P_{c,t-1}}{S_m}\right)} - e^{\left(\frac{-P_{c,t}}{S_m}\right)} \right) \tag{2}$$

where S_m is the maximum content of the interception store (m), $P_{c,t-1}$ is the cumulative precipitation since the start of rainfall, at the preceding time-step (m) and $P_{c,t}$ is cumulative precipitation (m) since the start of rainfall, at the current time-step. Potential infiltration is simulated with the Smith and Parlange equation (Smith and Parlange, 1978), additionally accounting for rock fragments in the soil (Woolisher *et al.*, 1990)

$$F = K \cdot \frac{e^{F_c/B}}{e^{F_c/B} - 1} \tag{3}$$

where K is the effective saturated hydraulic conductivity of the field (m per time-step), F_c is the cumulative infiltration since the start of rain (m), B is the saturation deficit parameter modified for rock fragments (m) and F is the potential amount of infiltration in a time-step (m per time-step). Water is routed through rill areas with the kinematic wave using the Manning equation (Li *et al.*, 1975; Chow *et al.*, 1988).

ENVIRONMENTAL MODELLING LANGUAGE FOR HYDROLOGICAL MODELLING

What is the best programming language to construct an event-based runoff model for the case study described above? For answering this question, languages can be compared by looking at the entities that are changed by the principal statements (operators) of the language and the type of functionality that is provided by the operators. Both the kind of entities and the functionality of the operators should (i) be compliant with the kind of objects that will be changed by the language and the changes that need to be made to them, and (ii) represent the thinking level of the user of the language. Table I gives the entities and the functionality of some widely used programming languages. It shows that some languages represent most aspects of the computer in their entities and operators, so called low-level languages, whereas others deal with entities and operators that are specific for a certain application field, so-called high-level languages. Assembly languages, being low-level languages, are not efficient for environmental model construction because virtually every aspect of the computer has to be defined in the program. System programming languages and generic scripting languages are at a higher level, because each operator represents several machine instructions that would need a block of program code if written in an assembly language.

The entities of system programming languages such as strings and floating points are not at the level of thinking of a hydrologist, neither do they represent the objects of study in hydrology. In a program for a hydrological model, entities are needed that represent hydrological objects such as landscapes (maps), below-surface composition (three-dimensional blocks), and time (time-series), whereas operations at these entities are needed that represent hydrological processes. To illustrate this, assume that the interception Equation (1) has to be implemented with a system programming language. Much program code would be needed for defining data structures and file formats of spatio-temporal data, iterations for defining the time, and a spatial 'multiply' operator operating for each map unit. It would be more convenient to use a programming language that sets the loop and the operations in just two statements:

$$\text{timer } 100$$

$$\text{save } I = P^{*}c$$

Table I. Entities and functionality of operators of some programming languages. Top: higher level languages, bottom: lower level languages

Language	Entity	Functionality of operators
Assembly languages	Bits	Changing bits
System programming languages (e.g. FORTRAN, C++)	Integers, floating points, arrays	Adding, summing, looping
General purpose scripting languages (e.g. Tcl/Tk, Python)	Strings, integers, arrays	Adding, summing, looping
Standard Query Language	Tabular data	Selecting data from a table, ordering
Technical computing languages (e.g. Splus, MATLAB)	Matrices, floating points	Matrix inversion, adding matrices, calculating statistics
Graphical modelling systems (e.g. ModelMaker, Stella)	Non-spatial states, fluxes	Fluxes between states
Environmental modelling languages (e.g., PCRaster, Idrisi)	Maps, time-series, blocks	Summing maps, iterating through time, topological links, transport of water, visualization

where 'timer' defines iterations saying that the statement below the timer has to be executed for 100 time-steps. The variables I, P and c are spatial (maps) or non-spatial entities defined implicitly in the language. The operator '*' multiplies two spatial entities and the 'save' operator means the result should be saved for each time-step. So the statements mean multiply map P with map c resulting in the map I, and do this for each time-step. Another example is the kinematic wave transport of water needed for the model. Kinematic wave transport with the Manning equation is a more complicated operation than multiplication but it is also generic and involves only a few input map entities (see Li *et al.*, 1975; Chow *et al.*, 1988)

$$Q_{\mathrm{t}} = \mathbf{f}(Dir, Q_{\mathrm{t}-1}, QIn, \alpha, \beta, T, D)$$

where $\mathbf{f}$ is the kinematic wave operator, Dir the map with directions of flow, $Q_{\mathrm{t}-1}$ the map with water discharge in direction Dir in the previous time-step (e.g. m^3/s), QIn the map with addition or subtraction of water to/from the flow (e.g. m^3/s), α the map with a coefficient (Li *et al.*, 1975; Chow *et al.*, 1988), β the map with the momentum or Boussinesq coefficient (Chow *et al.*, 1988), T the time-step (e.g. s), D the map with distance of flow to downstream unit in direction Dir and Q_{t} the map with water discharge in direction Dir at the current time-step (e.g. m^3/s).

In a system programming language, this operation takes several pages of code to be implemented because it is a spatial operation that needs a numerical solution of the kinematic wave equations. The higher level definition in an EML is one line of code, as one operation operating on spatial entities (maps), as shown above. This use of higher level entities and operators is the main concept of EML.

The practical application of this depends on the genericity of the entities and operators in the EML. A relatively limited number of entities and operators should support a wide range of models to be constructed, for many different hydrological situations. Simple operations such as loops and mathematical operations on maps make up the main part of most models and are generic. The same seems to hold for more complex operations such as kinematic wave transport or groundwater flow. Stable numerical solution techniques for most of these operations have been described in papers and standard textbooks (e.g. Bear and Verruijt, 1987; Chow *et al.*, 1988; Zheng, 1993; Olivera and Maidment, 1999) and can be regarded as methods that have proven their general application.

High-level statements for hydrological modelling can be made only with EMLs that have built-in knowledge of both spatial and temporal entities, supported by hydrological operators such as kinematic wave transport. Examples of such EMLs are GRASS (GRASS, 2001), PCRaster (Van Deursen, 1995; Wesseling *et al.*, 1996a; PCRaster, 2001), Simile (Simile, 2001) and concepts described by Takeyama (1997).

IMPLEMENTATION OF THE RUNOFF MODEL WITH PCRASTER

PCRaster has been used to illustrate how the runoff model can be implemented in an EML. Entities in PCRaster are series of raster maps for spatio-temporal attributes, time-series for temporal non-spatial data and look-up tables. Map entities are assigned a type according to their content in a hydrological–geographical context. Types used are Boolean, nominal and ordinal for classified data, scalar and directional for continuous data, and local drain direction representing drainage networks (Figure 3). The PCRaster language contains 125 operators operating on these entities. Operators included are non-spatial (point) operations, spatial operations (Burrough and McDonnell, 1998) and spatio-temporal time operations for reading and writing temporal data, e.g. hydrographs at specific locations. The concept of the language is similar to mathematical thinking and notation. As in mathematics, each operator in PCRaster solely affects the resulting variable of that operator and has no side effects on other variables in the program, which might be the case in system programming languages. Additionally, the syntax obeys mathematical notation. For example, creating a map containing for each cell the total amount of a variable in its upstream area is done with the 'catchmenttotal' operator of PCRaster. For infiltration:

$$\text{report } IUpstreamArea = \text{catchmenttotal}(I, Ldd);$$

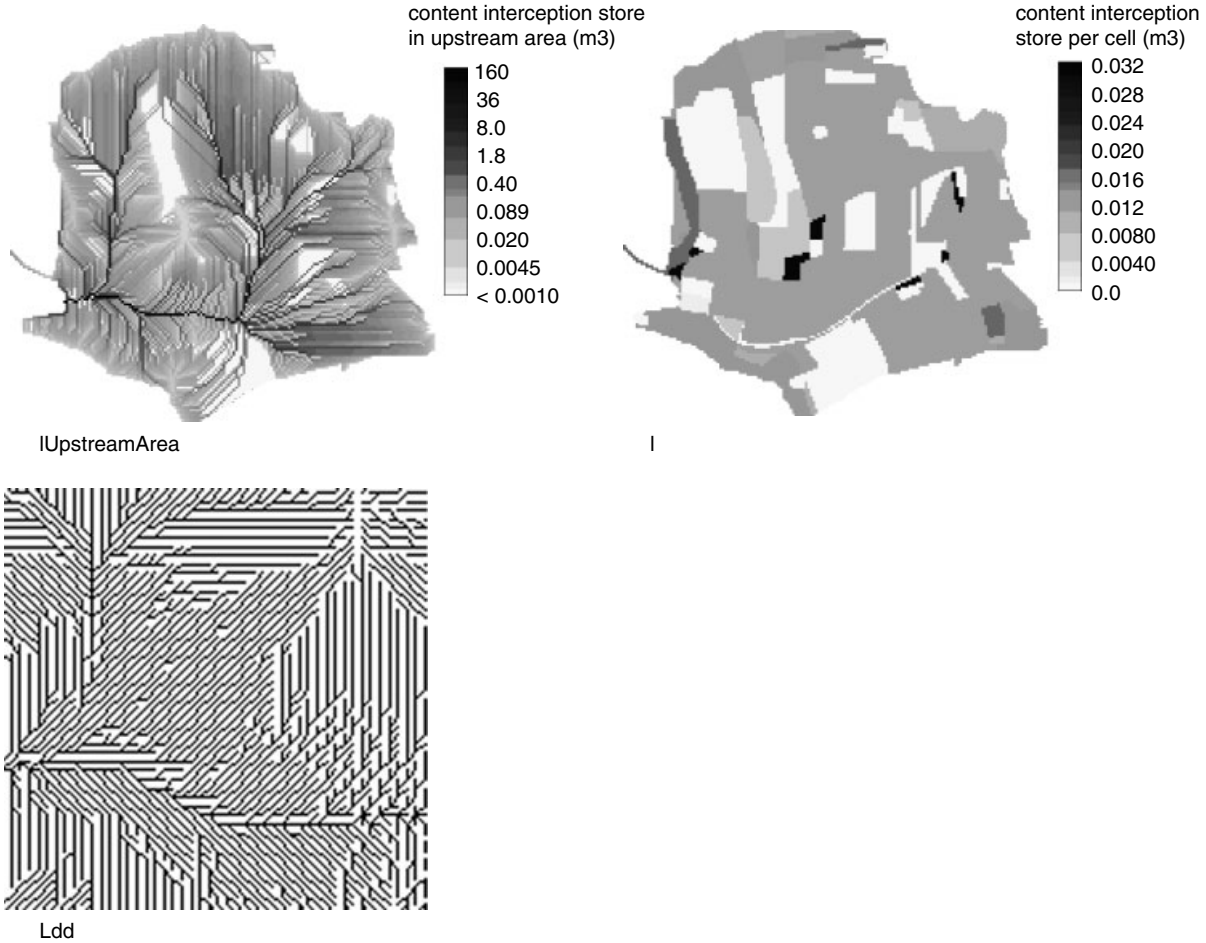

Figure 3. Input and output maps of *IUpstreamArea* = catchmenttotal(*I*, *Ldd*); *IUpstreamArea*, infiltration in upstream area; *I*, infiltration per cell; *Ldd*, part of the local drain direction map

where the 'catchmenttotal' operator has two inputs, the map *I* with the amount of infiltration for each cell and the map *Ldd*, the local drain direction map defining the drainage pattern. The operator generates a new map (here named *IUpstreamArea*) containing for each cell the total amount of infiltration in its catchment (Figure 3). In the implementation of the runoff model, budget checks at the catchment scale are performed by applying this operation on all water fluxes (e.g. precipitation, infiltration, surface storage) and summing results.

A PCRaster program (script) is structured in different sections (Table II). The binding is a list of program variables linked to filenames. The initial section initializes the model by setting initial states and constant values of variables using the PCRaster operators. In the case study model, operations in this section calculate constant parameters such as channel dimensions and roughness, interception parameters, infiltration parameters and initial states such as soil-water content. The dynamic section is an iterative sequential section and loops for the number of time-steps defined in the timer section. It contains operations for the temporal behaviour of the model: interception, infiltration, surface storage and kinematic wave transport. The result of each separate operation can be saved with the report keyword for each time-step or a selection of time-steps.

Database and visualization software, for spatio-temporal data using the file formats of the PCRaster language is integrated with PCRaster in one system. The case study model uses this software for scenario studies in

Table II. Program script of the model, text following # symbol are remarks. A selection of code is shown, code left out is indicated with '...'. Text in blocks describes left out or partly shown program code

BINDING

```
binding
 # time of one timestep (s)
 T = scalar (5);
 # map with locations of discharge measurements
 Flumes = report.map;
 ...
```

Links program variable names to input and output maps, timeseries, and tables

INITIAL

```
initial
 ...
```

interception parameters

```
 # nr of channels per metre, wheel tracks; ploughing
 report ChanPMWh = lookupscalar (ChanPMWhTBL, Landuse);
 report ChanPMPl = lookupscalar (ChanPMPlTBL, Landuse);
 ...
```

parameters for runoff routing

```
 ...
```

infiltration parameters and initial soil moisture. interrill & rill areas

```
 ...
```

surface storage parameters. interrill & rill areas

TIMER

```
timer
 # number of timesteps
 1 8500 1;
```

number of timesteps (iterations) for dynamic section

```
 # report at every timestep
 all = 1 + 1..endtime;
 # report at first and each 50th timestep
 anim = 1,50 + 50..endtime;
 ...
```

time-steps for which results are saved by report functions in the dynamic section

DYNAMIC

```
dynamic

 # rain (m/T)
 Pr = timeinputscalar (PTSS, Clone);
```

precipitation, reading rain time series

```
 # intercepted water (m/T)
 report (anim) Int = Pr*COV;
 # amount in interception store (m)
 ICSt = ICSStM* (1 - exp (-PCum/ICStM));
 ...
```

interception

```
 ...
```

infiltration and surface storage, interrill areas

```
 ...
```

infiltration and surface storage, rill areas

```
 ...
 # water added to the streamflow (m3/s)
 QIn = q*A_R/(T*DCL);
 # discharge (m3/s)
 report (all) Q = kinematic (Ldd, Q, QIn, Alpha, Beta, T, DCL);
 # simulated hydrograph (m3/s) at measurement locs
 report HydrographTss = timeoutput (Q,Flumes);
 # water depth (m), trapezoidal channels, abc formula
 H = (-NBw + sqrt (NBwSq + ZNFour*Alpha*(Q**Beta)))/ZNTwo;
 # wetted perimeter (m), multiple channels per cell
 P = NrCh*(Bw + 2*H*SqrtOnePlusSqrZ);
 # alpha
 report (anim) Alpha = AlpTerm*P**AlpPow;
 ...
```

routing surface water in rill areas

```
 # cumulative rill actual infiltration in upstream area (m3)
 FcSub_R = catchmenttotal(FcACCumV_R,Ldd);
 ...
 # total budget (rain minus sum of all fluxes should be 0)
 BudTot = PSub - (ICSub + FcSub_I + FcSub_R + DStSumb_I + DStSub_R + QrCum);
```

budget checks

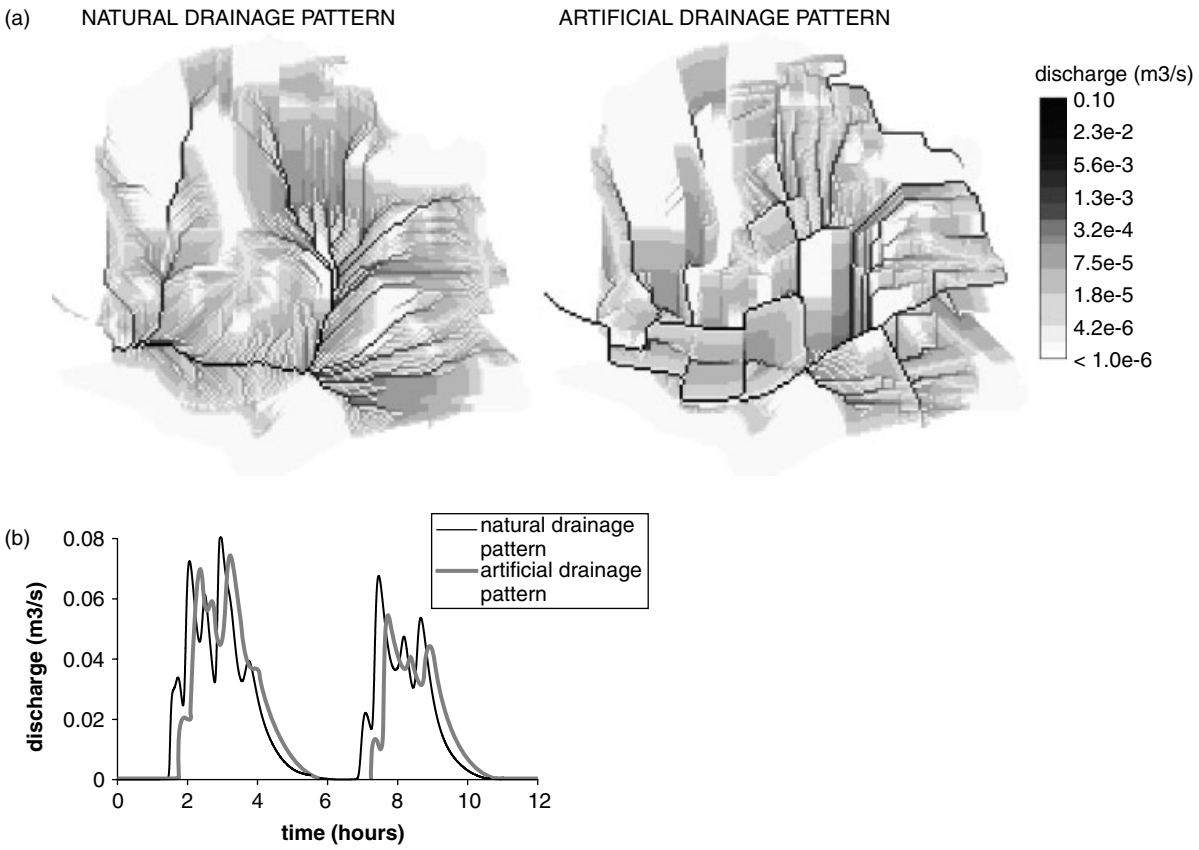

Figure 4. Runoff for one rain event, effect of drainage pattern on discharge. (a) Runoff after 3 h; left, natural drainage pattern; right, artificial drainage pattern. (b) Runoff at outflow point on the left-hand side of maps, for a natural drainage pattern and an artificial drainage pattern

cultivated land in the Ouvèze River basin (Southern France). The database consists of rain time-series files, 10 maps and 35 look-up tables that are directly read by the EML program. PCRaster supports prompt visualization of its entities with map and time-series display programs. Model results were significantly different for different patterns of runoff pathways, as Figure 4 shows (cf. Van Dijck, 2000). An extensive description of the model is available at WWW http://www.geog.uu.nl/pcraster/runoff/.

EVALUATION OF ENVIRONMENTAL MODELLING LANGUAGES

Based on the example of implementing the runoff model with PCRaster, it is possible to make a judgement about the usefulness of such an EML for hydrological model construction. This assessment is based on the requirements listed in an earlier section and a comparison is made with system programming. Table III provides a judgement based on the discussion below.

Level of thinking of hydrologists

Unlike system programming languages using entities and operators at the level of a computer, the EML uses entities and operators representing hydrological attributes, dimensions and processes. In addition, hydrologists are familiar with the mathematical notation used. This higher level of thinking of EML opens two possibilities. Unlike system programming languages, the EML can quickly teach environmental researchers without

Table III. Fulfilment of requirements for distributed hydrological modelling by system programming languages and PCRaster: +, mainly fulfilled; +/− partly fulfilled; −, not fulfilled

Requirement	System programming languages	PCRaster environmental modelling language
Level of thinking of hydrologists	−	+
Reuseable	+/−	+
Generic	+	−
No technical computer details	−	+
Short Development time	−	+
Minimizing programming errors	+/−	+/−
High performance	+	+/−
Easy to learn for hydrologists	−	+

specialist programming knowledge to develop models. Additionally models can be easily worked on by different researchers because they can read each other's programs. This is more difficult for system programs that generally can be read only by the specialist programmer in a research team.

Reuse of program code

The EML consists of higher level generic operators simulating hydrological processes that are combined to program the model. The development of these operators has been done in a system programming language by specialist programmers. So, from the technical point of view, combining operators in an EML means gluing together and reusing the program code that is behind these operators. The case study illustrates that this reuse is large. The model in the PCRaster language is only 340 lines of code whereas the amount of C++ code behind the PCRaster operators used is at least 15 000 lines.

Combining different blocks of code in a system programming language (e.g. Leavesley *et al.*, 1998) is in principle also possible, but needs the use of a style guide, which should be followed exactly by the programmer. Instead, an EML program implicitly provides standardization of the components combined, such as standard discretization of time and space in standard file formats. This implicit standardization allows automatic checks performed by the EML if operators are combined.

Generic approach to common problems

As noted in the introduction, an EML captures generic aspects of the hydrological processes in a standard set of operators. The PCRaster language, for instance, provides a means to write models for other fields of hydrology and geomorphology (Van Deursen and Kwadijk, 1993; Eleveld, 1999; De Roo *et al.*, 2000), and fields such as radio ecology (Van der Perk *et al.*, 2000), plant ecology (Van Deursen and Heil, 1993; Kessel, 1999), ornithology (Van Langevelde, 2000) and veterinary science (Brama *et al.*, 2000). Development of distributed models in PCRaster allows integrated modelling of (sub-) systems of the environment. Integration of different spatial–temporal processes, whether human, hydrological, ecological, geomorphological or toxicological, is possible in one orderly computer program (e.g. Van der Perk *et al.*, 2000).

A major restriction of EML is that the model builder is restricted by the concepts of the language and the operators provided. The PCRaster language, for instance, can be regarded as generic for simulating hydrological processes in two-dimensional space, although the user has to submit him or herself to the raster based approach in modelling. In addition, the current language has many restrictions that are not shown by our case study. It does not support three-dimensional spatial entities and operations such as groundwater flow, neither does it include standard functionality for stochastic modelling. This also is not supported by other EMLs. As the development of EML still continues, it can be expected that such restrictions will be solved in

the next 10 years, because concepts for these kinds of extensions exist (e.g. Karssenberg *et al*., 2000; Pebesma *et al*., 2000). Even with these additions, however, there always will be models that cannot be built with an EML, simply because the set of operators provided is not extensive. To solve this problem, users need to be able to program their own functions operating on maps in a system programming language, which can be made available as a standard operator in the EML. This increases the application range of the language. This plugging in of user made operators is already possible in systems such as GRASS and PCRaster.

No technical computer details

As EMLs are high-level languages, technical computer details do not need to be defined in an EML program. For instance, storing maps for each time-step is done by preceding a standard statement with the 'report' keyword in PCRaster, something that would take more code and format definitions if a system programming language is used.

Short development time

Evaluating different model structures and finding the optimal model for a specific research project is possible with EML, because alternative models can be built by recombining the standard operators of the language. Immediate post-modelling visualization of model results is possible with the standard visualization routines operating on the entities (e.g. maps) used in the language. As a result, models can be reconstructed, run and visualized in an almost interactive way, where it is easy to evaluate the model results for different alternative model structures. System programming languages do not support this interactive approach because these do not provide the scripting framework of EML to recombine and glue together operations. The short development time is illustrated by our rainfall-runoff model, which took one environmental researcher without specialist programming knowledge two weeks to program in the PCRaster language, with testing. Including and testing different interception equations and removing the interrill–rill process description for another modelling study took us 2 days. These development times are short compared with model development in a system programming language. This difference is caused mainly by the high-level character of the EML. Operators tested by developers and users of the EML are applied that would take weeks to be developed from scratch in a system programming language.

Minimising programming errors

Methods are embodied with EML for reducing programming errors that are not in system programming languages. By programming in an EML, high-level operators are combined that have been developed by professional programmers and tested in past research projects. For instance, the numerical solution implemented in the kinematic wave operator of PCRaster has been improved several times throughout the past 5 years, resulting in a stable, reliable operation with generic application. Someone who writes a model using this operator implements 5 years of research experience in the program. The data typing mechanism in PCRaster prevents the user from carrying out nonsense operations, because each operator is checked on its input and output. In preventing errors, it is important that the programmer has a good overview of the complete program and its inputs and outputs, which is guaranteed by the relatively short PCRaster scripts. In addition, intermediate model results in a scripting program can easily be stored on hard disk, by adding a save operator to a line, and these can be visualized with the embedded visualization tools. So models can be checked step by step by reporting such intermediate results.

This immediate visualization is the main debugging mechanism provided. Environmental modelling languages do not come with an automatic debugger, which is provided by system programming languages. This is a major disadvantage of EML, and it would be a useful addition, especially if it also provides an automatic mechanism to check budgets in hydrological models. Currently, the modeller has to define budget checks explicitly in the environmental modelling program. This is quite well possible using the standard

operators, but may involve quite an amount of work. The case study model includes 40 lines of code for checking all budgets.

High performance

Programs written by professional programmers in system programming languages will usually have shorter execution times than those written in EML. This mainly is because EMLs are scripting languages that have to perform more run-time checks than system programming software (Ousterhout, 1998). In addition, programs written in system programming languages can be optimized, whereas EMLs do not have many possibilities for optimization because fixed operators are glued together. On the other hand, the performance of an environmental modelling application tends to be dominated by the performance of the separate operators, which typically are implemented in a system programming language by professional programmers. This means that in some cases an EML program may be almost equally fast as a system programming language program. Still, EML models generally will be slower than models developed in system programming languages. Optimization of a system programming language version of a model developed originally in PCRaster decreased run times by a factor of eight (Wesseling *et al*., 1996b).

Easy to learn for hydrologists

An EML can be easily learned because the level of thinking of the language corresponds to the way of thinking of hydrologists (Karssenberg *et al*., 2001). The functionality of each of the operators in the language can be taught by using these operators from the command line, without using the iterative program. The next step, modelling in time, is to combine the operators in a program. By following this learning path, scientists can learn to use these languages in one or two weeks. This is a short learning curve compared with system programming languages, because for using these, understanding is needed of computer details.

FUTURE IMPLICATIONS FOR HYDROLOGICAL MODELLING

As EMLs are easy to learn for people without knowledge of programming, the threshold is low for hydrologists who wish to use them. Many researchers, however, will still prefer system programming languages. The choice of the programming language will depend mainly on the type of model that needs to be constructed. It can be expected that development of existing models will be done in the programming language (mostly a system programming language) in which these were developed originally, because rewriting these models in an EML would be inefficient for developers feeling at ease with the language they have always been using, and there is not much reason for rewriting these models in a different language anyway. Also for new models, system programming languages can be more efficient than EML. This holds mainly when performance of the model is important or when models need to be developed with concepts and functions that are not supported by an EML, although most EML come with a facility to plug in new operators, as noted above.

The strongest aspect of EML, and the main reason why they will have impact on future hydrological research, is their ease of use for constructing new models and to quickly modify existing models. In addition to currently existing standard models with fixed process equations, models worked on by many people will be developed that are under continuous change, because EMLs provide code that can be read easily and changed by many people who wish to tailor models to their research. Working on a model with several people is also possible with system programming languages, but these languages do not provide an explicit standardization of how to represent hydrological processes in computer code. Such standardization is provided by EML.

Also, EMLs are an extremely useful tool in the process of finding the optimal model, represented by the model development cycle (Figure 1), because modification of the model takes little time, and multiple models can be tried out in a certain research study. This opens the possibility to tailor a hydrological model to the available input data and aims of a modelling study. Such model adjustment is also possible using a system programming language, although it would take much more time.

CONCLUSIONS

Environmental modelling programming languages are conceptually attractive for hydrological model construction because they use entities and operators pitched at the level of thinking needed for writing numerical hydrological models. The case study with PCRaster shows that, compared with system programming, existing EMLs are better in their reusability of program code, lack of technical details in the program, short development time and learnability. They are equally good or worse than system programming languages in minimizing programming errors, generic application and performance. In future, it is expected that the application of system programming languages will move to more specialized models, whereas EML will be used to construct continuously evolving models where tailoring to study aims and field data is important.

ACKNOWLEDGEMENTS

The runoff modelling study was supported by the European Centre for Geomorphological Hazards. The case study runoff model was developed by the author in cooperation with Simone van Dijck (Utrecht University) to whom thanks are extended for cooperation and the data set. Peter Burrough, Thom Bogaard, Edzer Pebesma, Kor de Jong (Utrecht University), Marc Bierkens (Alterra, The Netherlands), Willem van Deursen and Cees Wesseling (PCRaster Environmental Software) are acknowledged for comments on the draft manuscript.

REFERENCES

Abbott MB, Bathurst JC, Cunge JA, O'Connel PE, Rasmussen J. 1986a. An introduction to the European Hydrological System—Systeme Hydrologique 'SHE', 1: history and philosophy of a physically-based distributed modelling system. *Journal of Hydrology* **87**: 45–59.

Abbott MB, Bathurst JC, Cunge JA, O'Connel PE, Rasmussen J. 1986b. An introduction to the European Hydrological System—Systeme Hydrologique 'SHE', 2: structure of a physically-based distributed modelling system. *Journal of Hydrology* **87**: 61–77.

AQUA3D. 2001. AQUA3D, *Groundwater Flow and Contaminant Transport Model*. Vatnaskil Consulting Engineers. http://www.vatnaskil.is/aqua3d.htm (Access date: 02/10/2001.)

Bear J, Verruijt A. 1987. *Modelling Groundwater Flow and Pollution. Theory and Applications of Transport in Porous Media*. Reidel: Dordrecht, 414.

Beven KJ (ed.). 1997. *Distributed Modelling in Hydrology: Applications of TOPMODEL.* Wiley: Chichester.

Brama PAJ, Tekoppelle JM, Bank RA, Karssenberg D, Barneveld A, Van Weerden PR. 2000. Topographical mapping of biochemical properties of articular cartilage in the equine fetlock joint. *Equine Veterinary Journal* **32**(1): 19–26.

Burrough PA. 1996. Opportunities and limitations of GIS-based modeling of solute transport at the regional scale. In *Application of GIS to the Modeling of Non-Point Source Pollutants in the Vadose Zone*, Corwin DL, Loague K (eds). American Society of Agronomy.

Burrough PA, McDonnell RA. 1998. *Principles of Geographical Information Systems*. Oxford University Press: Oxford.

Campbell AM. 1985. Discussion on the paper 'Groundwater modelling without special programs', by T. N. Olsthoorn. *Ground Water* **23**: 236–237.

Chow VT, Maidment DR, Mays LW. 1988. *Applied Hydrology*. McGraw-Hill: New York.

De Roo APJ, Wesseling CG, Van Deursen WPA. 2000. Physically based river basin modelling within a GIS; the LISFLOOD model. *Hydrological Processes* **14**: 1981–1992.

Donnelly-Makowecki LM, Moore RD. 1999. Hierarchical testing of three rainfall-runoff models in small forested catchments. *Journal of Hydrology* **219**: 136–152.

Eleveld M. 1999. *Exploring coastal morphodynamics of Ameland (the Netherlands) with remote sensing monitoring techniques and dynamic modelling in GIS*. PhD thesis, Universiteit van Amsterdam.

ESRI. 2001. *Environmental Systems Research Institute*. Info at: http://www.esri.com/ (Access date: 19/09/2001.)

GMS. 1998. *Groundwater Modelling System,* version 2.1. *Reference Manual*. Engineering Computer Graphics Laboratory (ECGL), Brigham Young University: Provo, Utah, USA. Info at: http://www.emrl.byu.edu/ (Access date: 02/10/2001.)

Grayson RB, Moore ID, McMahon TA. 1992. Physically based hydrologic modelling 1. a terrain-based model for investigative purposes. *Water Resources Research* **28**(10): 2639–2658.

GRASS. 2001. http://www.geog.uni-hannover.de/grass/ (Access date: 11/09/2001.)

Harbaugh AW, McDonald MG. 1996. *User's Documentation for MODFLOW-96, an Update to the U.S. Geological Survey Modular Finite-difference Ground-water Flow Model*. U.S. Geological Survey: Denver, Colorado, USA.

Highman B. 1967. *A Comparative Study of Programming Languages*. Computer Monographs 2. MacDonald: London; Elsevier: New York.

Karssenberg D, de Jong K, Burrough PA. 2000. A prototype computer language for environmental modeling in the temporal, 3D spatial and stochastic dimension. In *Proceedings of the 4th International Conference on Integrating GIS and Environmental Modeling*, 2–8 September, Banff, Canada. http://www.colorado.edu/research/cires/banff (Access date: 14/11/2000.)

Karssenberg D, Burrough PA, Sluiter R, de Jong K. 2001. The PCRaster software and course materials for teaching numerical modelling in the environmental sciences. *Transactions in GIS* **5**(2): 99–110.

Kessel G. 1999. Biological control of *Botrytis* spp. by *Ulocladium atrum*, an ecological analysis. PhD thesis, Wageningen University.
Leavesley GH, Restrepo PJ, Markstrom SL, Dixon M, Stannard LG. 1998. *The Modular Modelling System (MMS): User's Manual*. Open-File Report 96-151, U.S. Geological Survey: Denver, Colorado, USA. Available at: http://wwwbrr.cr.usgs.gov/mms (Access date: 14/11/2000.)
Li R-M, Simons DB, Stevens MA. 1975. Nonlinear kinematic wave approximation for water routing. *Water Resources Research* **11**(2): 245–252.
Mackay DS, Ban LE. 1997. Forest ecosystem processes at the watershed scale: dynamic coupling of distributed hydrology and canopy growth. *Hydrological Processes* **11**: 1197–1217.
MATLAB. 2001. Info at: http://www.mathworks.com/ (Access date: 02/09/2001.)
Merriam RA. 1973. Fog drip from artificial leaves in a fog wind tunnel. *Water Resources Research* **9**: 1591–1598.
ModelMaker. 2001. Info at: http://www.modelkinetix.com/modelmaker/ (Access date: 02/09/2001.)
Morgan RPC, Quinton JN, Smith RE, Govers G, Poesen JWA, Auerswald K, Chisci G, Torri D, Styczen ME. 1998. The European soil erosion model (EUROSEM): a dynamic approach for predicting sediment transport from fields and small catchments. *Earth Surface Processes and Landforms* **23**: 527–544.
Olivera F, Maidment D. 1999. Geographic information systems (GIS)-based spatially distributed model for runoff routing. *Water Resources Research* **35**(4): 1155–1164.
Olsthoorn TN. 1998. *Groundwater Modelling: Calibration and the Use of Spreadsheets*. Delft University Press: Delft.
Ousterhout JK. 1998. Scripting: higher level programming for the 21st Century. *Computer* **31**(3): 23–30.
Pebesma EJ, Karssenberg D, de Jong K. 2000. The stochastic dimension in a dynamic GIS. In *Proceedings of Compstat 2000, 14th Conference of the International Association for Statistical Computing*, 21–25 August, Utrecht.
PCRaster. 2001. Information available at http://www.geog.uu.nl/pcraster (Access date: 02/09/2001.)
Smith RE, Parlange J-Y. 1978. A parameter-efficient hydrologic infiltration model. *Water Resources Research* **14**(3): 533–538.
STELLA. 2001. Info at: http://www.hps-inc.com/edu/stella/stella.htm (Access date: 02/09/2001.)
Simile. 2001. *A Modelling Environment for Ecological and Environmental Modelling*. Available at: http://www.ierm.ed.ac.uk/simile/index.html (Access date: 11/09/2001.)
Takeyama M. 1997. Building spatial models within GIS through Geo-Algebra. *Transactions in GIS* **2**(3): 245–256.
Tucker G, Gasparini N, Bras RL, Rybarczyk S. 1999. An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks. In *Proceedings of GeoComputation 99, 4th International GeoComputation Conference*, Fredericksburg, USA, 25–28 July. http://www.geovista.psu.edu/geocomp/geocomp99 (Access date: 14/11/2000.)
Van der Perk M. 1997. Effect of model structure on the accuracy and uncertainty of results from water quality models. *Hydrological Processes* **11**: 227–239.
Van der Perk M, Lev T, Gillet AG, Absalom JP, Burrough PA, Crout NMJ, Garger EK, Semiochkina N, Stephanishin YV, Voigt G. 2000. Spatial modelling of transfer of long-lived radionuclides from soil to agricultural products in the Chernigov region, Ukraine. *Ecological Modelling* **128**: 35–50.
Van Deursen WPA. 1995. *Geographical information systems and dynamic models*. PhD thesis, Utrecht University, NGS Publication 190; 198 pp.
Van Deursen WPA, Kwadijk JCJ. 1993. RHINEFLOW: an integrated GIS water balance model for the river Rhine. In *Applications of Geographic Information Systems in Hydrology and Water Resources Management, HydroGIS 1993*, Kovar K, Nachtnebel HP (eds). IAHS Publication **211**; International Association of Hydrological Processes: Wallingford; 507–518.
Van Deursen WPA, Heil GW. 1993. Analysis of heathland dynamics using a spatial distributed GIS model. *Scripta Geobotanica* **21**: 17–28.
Van Deursen WPA, Wesseling CG, Karssenberg D. 2000. How do we gain control over GIS technology? In *Proceedings of the 4th International Conference on Integrating GIS and Environmental Modeling*, 2–8 September, Banff, Canada. http://www.colorado.edu/research/cires/banff (Access date: 14/11/2000.)
Van Dijck SJE. 2000. *Effects of agricultural land use on surface runoff and erosion in a Mediterranean area*. PhD thesis, Utrecht University, NGS Publication 265; 246.
Van Langevelde F. 2000. Scale of habitat connectivity and colonization in fragmented nuthatch populations. *Ecography* **23**: 614–622.
Wesseling CG, Karssenberg D, Van Deursen WPA, Burrough PA. 1996a. Integrating dynamic environmental models in GIS: the development of a dynamic modelling language. *Transactions in GIS* **1**: 40–48.
Wesseling CG, Van Deursen WPA, Burrough PA. 1996b. A spatial modelling language that unifies dynamic environmental models and GIS. In *Proceedings, Third International Conference/Workshop on Integrating GIS and Environmental Modelling*, Santa Fe, NM, 21–26 January, National Center for Geographic Information and Analysis: Santa Barbara, CA. http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/main.html (Access date: 14/11/2000.).
Woolisher DA, Smith RE, Goodrich DC. 1990. *KINEROS, a Kinematic Runoff and Erosion Model: Documentation and User Manual*: Tucson, Arizona, USA, 130.
Zheng C. 1990. *MT3D, a Modular Three-dimensional Transport Model for Simulation of Advection, Dispersion and Chemical Reaction of Contaminants in Groundwater Systems*. United States Environmental Protection Agency: Aola, Oklahoma, USA.
Zheng C. 1993. Extension of the method of characteristics for simulation of solute transport in three dimensions. *Ground Water* **31**(3): 456–465.

