Environmental Modelling & Software xxx (2014) 1–8

Contents lists available at ScienceDirect

Environmental Modelling & Software

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

Serving many at once: How a database approach can create unity in dynamical ecosystem modelling☆

Wolf M. Mooij[a,b,*], Robert J. Brederveld[c], Jeroen J.M. de Klein[b], Don L. DeAngelis[d], Andrea S. Downing[e], Michiel Faber[c], Daan J. Gerla[f,p], Matthew R. Hipsey[g], Jochem 't Hoen[b], Jan H. Janse[h], Annette B.G. Janssen[a,b], Michel Jeuken[i], Bob W. Kooi[j], Betty Lischke[k], Thomas Petzoldt[l], Leo Postma[h], Sebastiaan A. Schep[c], Huub Scholten[m], Sven Teurlincx[a], Christophe Thiange[h], Dennis Trolle[n], Anne A. van Dam[o], Luuk P.A. van Gerven[a,b], Egbert H. van Nes[b], Jan J. Kuiper[a,b]

[a] *Department of Aquatic Ecology, Netherlands Institute of Ecology (NIOO-KNAW), P.O. Box 50, 6700 AB Wageningen, The Netherlands*
[b] *Aquatic Ecology and Water Quality Management Group, Department of Environmental Sciences, Wageningen University, P.O. Box 47, 6700 AA Wageningen, The Netherlands*
[c] *Witteveen+Bos, P.O. Box 233, 7400 AV Deventer, The Netherlands*
[d] *USGS/Biological Resources Division and Department of Biology, University of Miami, P.O. Box 249118, Coral Gables, FL 33124, USA*
[e] *Department Systems Ecology, Stockholm University, 10691 Stockholm, Sweden*
[f] *Department of Ecosystems, Institute for Marine Resource and Ecosystem Studies (IMARES), PO Box 167, 1790 AD Den Burg, The Netherlands*
[g] *University of Western Australia, School of Earth and Environment, Crawley, WA 6009, Australia*
[h] *PBL, Netherlands Environmental Assessment Agency, P.O. Box 303, 3720 AH Bilthoven, The Netherlands*
[i] *Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands*
[j] *Department of Theoretical Biology, Faculty of Earth and Life Sciences, VU University, De Boelelaan 1085, 1081 HV Amsterdam, The Netherlands*
[k] *Department of Ecology and Ecosystem Modelling, Institute of Biochemistry and Biology, University of Potsdam, Am Neuen Palais 10, 14469 Potsdam, Germany*
[l] *Faculty of Environmental Sciences Institute of Hydrobiology, Technische Universität Dresden, 01062 Dresden, Germany*
[m] *Information Technology Group, Department of Social Sciences, Wageningen University, P.O. Box 8130, 6700 EW Wageningen, The Netherlands*
[n] *Department of Bioscience, Aarhus University, Vejlsøvej 25, 8600 Silkeborg, Denmark*
[o] *UNESCO-IHE Institute of Water Education, 2601 DA Delft, The Netherlands*
[p] *Department of Ecosystem Studies, Royal Netherlands Institute for Sea Research (NIOZ), PO Box 140, 4400 AC Yerseke, The Netherlands*

ARTICLE INFO

Article history:
Received 30 June 2013
Received in revised form
1 April 2014
Accepted 3 April 2014
Available online xxx

Keywords:
Modelling framework
Programming language
Differential equation
Community-based modelling
Database approach to modelling
DATM

ABSTRACT

Simulation modelling in ecology is a field that is becoming increasingly compartmentalized. Here we propose a Database Approach To Modelling (DATM) to create unity in dynamical ecosystem modelling with differential equations. In this approach the storage of ecological knowledge is independent of the language and platform in which the model will be run. To create an instance of the model, the information in the database is translated and augmented with the language and platform specifics. This process is automated so that a new instance can be created each time the database is updated. We describe the approach using the simple Lotka–Volterra model and the complex ecosystem model for shallow lakes PCLake, which we automatically implement in the frameworks OSIRIS, GRIND for MATLAB, ACSL, R, DUFLOW and DELWAQ. A clear advantage of working in a database is the overview it provides. The simplicity of the approach only adds to its elegance.

1. Introduction

Since the onset of ecological simulation modelling based on differential equations – in the sixties and seventies of the last century – attempts have been made to bring conceptual unity through the development of modelling frameworks. In the field of

☆ Thematic Issue on the Aquatic Ecosystem Models.
* Corresponding author. Department of Aquatic Ecology, Netherlands Institute of Ecology (NIOO-KNAW), P.O. Box 50, 6700 AB Wageningen, The Netherlands. Tel.: +31 6 3177 1708.
E-mail address: w.mooij@nioo.knaw.nl (W.M. Mooij).

http://dx.doi.org/10.1016/j.envsoft.2014.04.004

aquatic ecology, such frameworks include the widely used DELWAQ – a library of water quality and ecology models developed by Delft Hydraulics (Delft Hydraulics, 1995; Deltares, 2013), as well as the Computational Aquatic Ecosystem Dynamics Model (CAEDYM) – a library of ecological process sub-models (Hipsey et al., 2007), AQUASIM (Reichert, 1994), the Dutch Waterboards' DUFLOW framework (Spaans et al., 1989) and the recently developed FABM – Framework for Aquatic Biogeochemical Models (http://fabm.sourceforge.net). Each of these frameworks is internally consistent, intuitive and well suited to answer the ecological questions it was designed for (Clemmens et al., 1993; Gal et al., 2004), and all are based on the same basic mathematical principles underlying the differential equations. Nonetheless, because these frameworks were developed independently, they all have their own sets of implementation requirements, language and coding specifications, spatial configuration options as well as boundary conditions and forcing function specifications, etc.

A user must therefore invest a considerable amount of effort to master any given framework, which in turn reduces the number of frameworks that any single user can master. The choice of framework to be used for any given project is thus primarily based on its availability, owned licenses, user experience and developer familiarity. This in turn leads to models being locked into their given frameworks, a narrowing-down of scientific expertise to the framework-scale and to the proverbial 're-invention of the wheel' – i.e., the inefficient redevelopment of existing tools for each framework, rather than a more productive cross-pollination of approaches to analyze models across frameworks, institutions, disciplines and scientists (Leavesley et al., 2002; Mooij et al., 2010; Trolle et al., 2012). We are confronted with the paradoxical situation that, while there is unity within each framework, there is no unity at the level of the ecological models.

Here we propose a method to bring unity at the level of the ecological module, with the idea that many of the existing frameworks will continue to coexist, and that, taken together, they provide the user with a wide and rich array of tools for model analysis. We coin this method a 'Database Approach To Modelling' (DATM). We developed this approach for the ecosystem model for shallow lakes PCLake, and its twin model for linear waters PCDitch. However, our approach is in no way limited to these models. In fact, it applies to all models based on differential equations and probably even beyond. We here show how one can automatically link these models to a wide variety of frameworks, including OSIRIS (Mooij and Boersma, 1996), GRIND for MATLAB (available on http://www.sparcs-center.org/grind.html), ACSL (Mitchell and Gauthier, 1976), R (R Development Core Team, 2008), DUFLOW (Spaans et al., 1989) and DELWAQ (Deltares, 2013). Note that the latter two frameworks are spatially explicit and therefore are formulated in terms of partial differential equations (PDE's), whereas implementations of an ecological model (e.g. PCLake) in the general purpose frameworks are a set of ordinary differential equations (ODE's). We will show that with DATM we can overcome this difference, and translate a single code either in a set of ODE's in a general purpose framework or as the ecological component of a set of PDE's in these spatially explicit frameworks. In the latter case, these ecological components are then merged by the frameworks with the advective and diffusive transport of matter to get the full PDE. Please note that in its current form, DATM does not provide the spatial configuration of the model, this has still to be entered at the level of the framework.

To explain the principles of DATM, we use as an example the classical Lotka–Volterra equations. These equations represent the earliest use of coupled differential equations in ecology (Lotka, 1920; Volterra, 1926, 1931). With this example, we show how knowledge of quite a few framework-specific details is necessary to implement even this simplest of models in some of the most widely used mathematical frameworks. From experience, we have learned how implementing more complex models in more specific frameworks takes a considerable effort, which is why we propose to automate this process: an essential component of DATM is the set of translators developed to automatically convert the database definitions of a given model into a working implementation in a specific framework. Conceptually, we argue that the overview and insight that arises when the model definition is stored in the database, conveniently displayed in tables and accessed through queries, facilitates model development and understanding.

2. Methods

DATM is based on the notion that ecological models are essentially rooted in mathematics. Here, we focus on models based on the mathematical concept of coupled differential equations. The dynamic systems represented by these equations have a universal mathematical notation. As an example, the Lotka–Volterra predator-prey equations can be read and understood by all in the following form:

$$\mathrm{d}V/\mathrm{d}t = r\,V - a\,V\,P \tag{1a}$$

$$\mathrm{d}P/\mathrm{d}t = a\,e\,V\,P - \mathrm{d}P \tag{1b}$$

with state variables V for prey and P for predator; parameters r for autonomous growth rate of the prey; a the attack rate of the predator on the prey, e the conversion efficiency of the predator and d the autonomous death rate of the predator. This system is in this form fully defined and ready for simulation for a given set of parameters r, a, e and d and initial conditions $V_{t=0}$ and $P_{t=0}$. Our central point is that this mathematical notation for complex simulation models is sufficient to achieve unity and transparency in ecological modelling.

As shown in the above example, the set of coupled Equations (1a) and (1b) must be augmented with information on the interpretation of the various identifiers that are used in the model. As a minimum description, the identifiers must belong to a certain class (e.g. state variable, parameter); represent a specific component of the system (e.g. prey, predator); have units (e.g. biomass, number of individuals), and (initial) values. In scientific papers that document smaller models, such as the Lotka–Volterra model, this information is often organized in tables, with either a shared table for all identifiers or separate tables per class of identifiers. Given the number of identifiers in the more complex water quality models, we choose to work with separate tables for each class of identifiers. For the Lotka–Volterra model such tables could look like (note the 's' prefix to identifiers of state variables):

Table 1
State variables.

Identifier	Description	Dimension	Initial value
sV	Prey density	Biomass V	(Some number)
sP	Predator density	Biomass P	(Some number)

for the states,

Table 2
Parameters.

Identifier	Description	Dimension	Value
r	Prey growth rate	Time^{-1}	(Some number)
a	Predator attack rate	Time^{-1} biomass P^{-1}	(Some number)
e	Predator efficiency	Biomass P biomass V^{-1}	(Some number)
d	Predator death rate	Time^{-1}	(Some number)

for the parameters and

Table 3
Derivatives.

Identifier	Description	Dimension	Equation
dV	Prey derivative	Biomass V time^{-1}	$\mathrm{d}V = r \cdot sV - a \cdot sV \cdot sP$
dP	Predator derivative	Biomass P time^{-1}	$\mathrm{d}P = a \cdot e \cdot sV \cdot sP - \mathrm{d} \cdot sP$

for the derivatives. Extra columns with additional information, such as the references for the parameter values, can be added, of course, until all relevant information is stored in the tables. We thus reach a full documentation of the model in a set of linked tables; i.e., in a database.

To create an instance of the model for a certain framework, the information in the database of Tables 1–3 is translated and augmented to meet the specifications of running it in the chosen framework. For instance, a running version of the above model (Fig. 1) can be obtained by producing code for MATLAB (Box 1), Mathematica (Box 2), or R (Box 3).

Box 1
Implementation of the Lotka–Volterra equations in MATLAB.

```
function LotkaVolterra_ode45

% set initial values
sV_0 = 10;
sP_0 = 10;

% integrate the model
options = odeset('RelTol', 0.0001, 'NonNegative', [1 2]);
[t, x] = ode45(@LotkaVolterra, [0 20], [sV_0 sP_0], options);

% show the results
plot(t, x);
legend('sV', 'sP');

% define the model
function dx = LotkaVolterra(t, x)

% set parameters
r = 1;
a = 0.05;
e = 0.4;
d = 0.5;

% copy x to states
sV = x(1);
sP = x(2);

% calculate derivatives
dV = r * sV - a * sV * sP;
dP = a * e * sV * sP - d * sP;

% copy derivatives to x
dx(1, 1) = dV;
dx(2, 1) = dP;
```

Note that each of these implementations needs information that controls the simulation such as the integration method and time step (t-int) and the time interval over which the model is run (t-end). This essential information is specified in an additional table in the database (Table 4).

Additionally, tables can be included that hold input time series data for forcing functions, or data for calibration or validation. Simultaneously with the translation of the model code, the data are translated to the format needed by the different frameworks.

To apply the approach, we implemented the Tables 1–4 in a Microsoft Excel Workbook as Worksheets (see Section S1 of the online supplementary material). We would like to stress that any program that can hold tables could be used. We chose Excel because it is widely available, and most people are familiar with it. Microsoft Access is an alternative that might provide a more rigid control of the database, but fewer people have experience with it. A freeware alternative would be LibreOffice, which also has the advantage of being easily portable to Mac, Linux and Windows.

Box 2
Implementation of the Lotka–Volterra equations in Mathematica.

```
(* define the model *)
ode = {
  sV'[t] == r sV[t] - a sV[t] sP[t],
  sP'[t] == a e sV[t] sP[t] - d sP[t]
};

(* set parameters *)
par = {r -> 1, a -> 0.05, e -> 0.4, d -> 0.5};

(* set initial values *)
ic = {sV[0] == 10, sP[0] == 10};

(* set run time *)
t0 = 0; t1 = 20;

(* integrate the model *)
eqns = ode~Join~ic;
sol = NDSolve[eqns /. par, {sV[t], sP[t]}, {t, t0, t1}];

(* show the results *)
Plot[{sV[t], sP[t]} /. sol, {t, t0, t1}]
```

Box 3
Implementation of the Lotka–Volterra equations in R.

```
# define the model
LotkaVolterra <- function(times, states, parameters){
  with(as.list(c(states,parameters)), {
    dV <- r * sV - a * sV * sP
    dP <- a * e * sV * sP - d * sP
    list(c(dV, dP))
  }
)}

# set parameters
parameters <- c(r = 1, a = 0.05, e = 0.4, d = 0.5)

# set initial values
states <- c(sV = 10, sP = 10)

# set run time
times <- seq(from = 0, to = 20)

# integrate the model
library("deSolve")
results <- ode(states, times, LotkaVolterra, parameters,
method="ode45")

# show the results
plot(results)
```

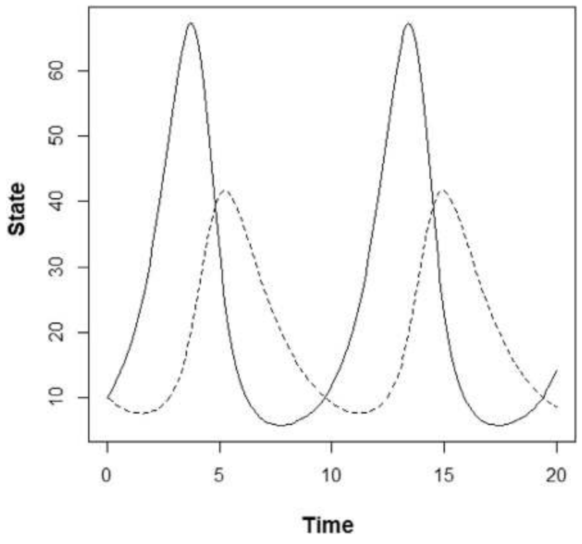

Fig. 1. Typical model output for the Lotka–Volterra example presented in Boxes 1–3. The solid line shows the dynamics of prey density *V*, the dashed line the dynamics of predator density *P*.

Using Excel Macros and Visual Basic for Applications (VBA), we wrote translators that turn the information provided in Tables 1–4 into the working scripts provided in Boxes 1–3 (the code of the translators can be found in Section S2 of the online supplementary material and the code they produce in Section S3 A–C of the online supplementary material). Again, these translators can be written in any language that easily handles tables, records, and text strings such as R, Python or PERL. We chose VBA because it is embedded in Excel. The validity of these translators can be checked by comparing the results of benchmark runs against each other. These not only show the (dis)similarity in model outcomes, but also give an indication of the performance of the model under study in each framework. Thereafter, the model can be analyzed with the tools provided by each framework (e.g. the "paranal" function for sensitivity analysis in GRIND for MATLAB). DATM therefore provides easy access to existing tools of analysis in various frameworks, without providing these tools itself.

We have applied the methodology described above to implement the ecosystem models for shallow lakes PCLake (Janse et al., 2008, 2010) and for shallow linear waters PCDitch (Janse, 1998; Van Liere et al., 2007) in the frameworks OSIRIS, ACSL, GRIND for MATLAB, R, DUFLOW and DELWAQ. PCLake and PCDitch are integrated ecological models to study the main nutrient and food web dynamics of shallow lakes and ditches in response to eutrophication and associated restoration measures (See Mooij et al., 2010 for a comparison with other water quality models). Both models are frequently used in both water quality management and for scientific investigations. For brevity, we will only refer to PCLake in the results, since its implementation is technically equivalent to that of PCDitch.

3. Results

PCLake is about two orders of magnitude more complex than the Lotka–Volterra model. It has 104 state variables and approximately 400 parameters. Instead of calculating the right hand sides of the differential equations directly, it uses near 1500 intermediate variables to calculate components that are used in the 104 differential equations. PCLake also includes a set of equations that are calculated before running the simulation to make sure that the initial values of the states obey certain basic biological rules (e.g. stoichiometric constraints) when initial values are provided only for dry-weight values but not for N and P. These equations also set the initial composition of the sediment. The PCLake database therefore consists of five instead of four tables: 1) Simulation information, 2) States, 3) Parameters, 4) Initial equations, 5) Dynamic equations (calculation of auxiliaries and derivatives). The last table could have been split into two tables but with experience we find that we get a better model-overview when auxiliaries and the derivatives are in a single table. We refer to Section S4 of the online supplementary material for the definition of each table of the PCLake implementation in DATM and for a comparison with the Lotka–Volterra example.

Tables 1–4 show the minimal record structure for each table in the Lotka–Volterra example. For PCLake in DATM, we added a column to each table to number the identifiers, and a column to provide additional information per identifier. The table approach also allows one to enter multiple input vectors for initial values of states and of parameters. By adding variables to the simulation table that specify which input vector is used in a given simulation, one can compare model runs for various initial values and/or parameter sets. This approach can be extended to the column in which the model equations are specified. Different columns then characterize multiple versions of the model in a single table. The version of the equations to be used can then be specified in the simulation table. This allows for a straightforward comparison of runs for different model equations and even for different model structures where, for example, certain state variables and associated fluxes are added or switched off. DATM thus facilitates sensitivity analyses on both parameters and model structure.

The Lotka–Volterra example only contains the addition (+), multiplication (*) and equality (=) mathematical operators, but more complex models can include power (e.g. ^), relational operators (e.g. >) and logical operators (e.g. AND), as well as conditional statements (e.g. IF-THEN-ELSE). Operators and statements have distinct implementations in the dominant multi-purpose computer

Table 4
Information controlling the simulation.

Model	Integration method	t-int	t-end
Lotka–Volterra	ode45	0.1	20

Table 5
Translations of conditional statements, logical operators and mathematical functions from the database to each of the six modelling platforms.

FRAMEWORK	OSIRIS	GRIND	ACSL	R	DUFLOW	DELWAQ
Language	C++	MATLAB	ACSL	R	DUPROL	FORTRAN
_IF_	(blank)	if	IF	if	if	if
THEN	?	(cr)	THEN (cr)	{ (cr)	{ (cr)	then (cr)
ELSEIF	:	(cr) elseif	(cr) ELSEIF	(cr) } else if	(cr) } else if	(cr) else if (cr)
ELSE	:	(cr) else (cr)	(cr) ELSE (cr)	(cr) } else { (cr)	(cr) } else { (cr)	(cr) else (cr)
ENDIF	(blank)	(cr) end	(cr) ENDIF	(cr) }	(cr) }	endif
_EQ_	==	==	.EQ.	==	==	==
_NE_	!=	~=	.NE.	!=	!=	/=
_GE_	>=	>=	.GE.	>=	>=	>=
_LT_	<	<	.LT.	<	<	<
_GT_	>	>	.GT.	>	>	>
_LE_	<=	<=	.LE.	<=	<=	<=
TRUE	1	true	.TRUE.	1	1	1
FALSE	0	false	.FALSE.	0	0	0
AND	&&	&&	.AND.	&&	&&	.and.
_OR_	||	||	.OR.	||	||	.or.
FLOOR	floor	floor	INT	floor	int	floor
COS	cos	cos	COS	cos	cos	cos
SIN	sin	sin	SIN	sin	sin	sin
TAN	tan	tan	TAN	tan	tan	tan
ACOS	acos	acos	ACOS	acos	acos	acos
ASIN	asin	asin	ASIN	asin	asin	asin
ATAN	atan	atan	ATAN	atan	atan	atan
EXP	Exp	exp	EXP	exp	exp	exp
MIN	Min	min	MIN	min	min	min
MAX	max	max	MAX	max	max	max
_LN_	log	ln	LOG	log	ln	log
POW	pow	(blank)	(blank)	(blank)	(blank)	(blank)
_^_	,	^	**	^	^	**

(blank) = no entry, (cr) = new line.

languages such as C++ and FORTRAN. The difference is usually in the syntax (e.g. '&&' in C++ is '.and.' in FORTRAN), though sometimes operators do not have their equivalent in all languages (e.g. the power-operator is missing in C++). Furthermore, some frameworks have their own computer languages, such as DUFLOW, where modules are written in the language DUPROL. Table 5 contains a complete list of translations used in PCLake and PCDitch.

All operators except '=', '+' and '*' and all standard mathematical functions are given a unique text-based identifier in the database. These unique identifiers of operators and functions are then translated into an automated search-and-replace operation. For this reason, a correct translation into any specific language can only be guaranteed if operators cannot be confused with parts of names of other identifiers. In the same way, the names of identifiers, state variables, parameters or intermediate variables must be completely unique, i.e. they should not be contained in the name of any other identifier. Each identifier in the database is therefore preceded and followed by a unique symbol. We propose to use the underscore, since it has no specific meaning in mathematics and enhances the readability of the equations.

The database format prescribes that all the right hand terms for a given identifier are given on a single line; we therefore used the following style:

```
left hand term = _IF_ condition 1 _THEN_ right hand
term 1 _ELSEIF_ condition 2 _THEN_ right hand term 2
_ELSE_ right hand term 3 _ENDIF_
```

For C-based languages, this can be easily translated into a conditional expression using the ternary operator "? :":

```
left hand term = condition 1 ? right hand term 1 :
condition 2 ? right hand term 2 : right hand term 3
```

For other languages it can be translated into the more traditional "IF-THEN-ELSE" construct:

```
IF condition 1 THEN
    left hand term = right hand term 1
ELSE
    IF condition 2 THEN
        left hand term = right hand term 2
    ELSE
        left hand term = right hand term 3
    ENDIF
ENDIF
```

Another small obstacle towards generality is the absence of a power operator in C-based languages. Power functions such a^b are entered in the database with a combination of both styles: _POW_ (a _^_ b), which can easily be translated to C as pow(a, b), or to FORTRAN as (a ** b) (note the essential parenthesis).

As demonstrated in the implementations of the Lotka–Volterra model in MATLAB, R and Mathematica, the model code is preceded and followed by certain statements that bridge the code defining the model *sensu stricto* and the framework. What information should be provided – or omitted – depends on the specific framework; some frameworks make use of a graphical user interface that is difficult to circumvent (e.g. DUFLOW). The spatial capabilities of DELWAQ and DUFLOW prescribe that the corresponding simple single cell modules for hydrology and transport available in PCLake should be excluded during translation, as these processes are taken care of by these frameworks. Note that the integration between the ODE process formulations provided by DATM and the PDE process formulations of the framework is taken care of by the framework. To enable integration with an existing water quality model, process modules formulated as ODE's can be stored in a repository in both DUFLOW and DELWAQ. The DATM translator simply adds another model to these repositories. For spatially-explicit frameworks that lack such build-in facilities for the incorporation of water quality models formulated as ODE's, a more customized integration is necessary, given that any framework should have some formal entry point for these equations. As of yet, however, we do not have experience with such frameworks. Some details about the richer structure of the implementation of PCLake (and PCDitch) in the different frameworks can be found in Section S5 of the online supplementary material.

After solving the inevitable errors that are reported by the compiler or interpreter, it is essential to check that the newly translated code functions correctly. An effective first step is to calculate the value of each identifier (all parameters, initial states, intermediate variables and derivatives) at $t = 0$ and compare these values with a control set. This dump output at $t = 0$ is also very useful in studying the main and side effects of changes to the code and is therefore a standard asset of the approach that we advocate.

Secondly, benchmark simulations of varying complexity reveal the proper functioning of conditional statements and forcing functions. This is clearly shown as we overlay time plots from two different frameworks (Fig. 2a, b). Of course, small differences remain because of machine rounding of errors and small differences arising from numerical integration. However, these differences are several orders of magnitude smaller than the ecological range of each state and therefore not visible when we plot the outcome of all frameworks for a given state against each other over this full range (Fig. 3). Such benchmark runs also demonstrate the runtime performance, which can be an important criterion for the choice of a framework. Obviously, one is limited in such runs to a model setup that can be handled by all the frameworks that participate in the test.

One should take into consideration that most platforms support different routines for numerical integration that do not need to be the same and thus influence both the accuracy of the model output and the runtime performance. Moreover, the difference between compiled languages (e.g. C++, FORTRAN) and scripting languages (e.g. R, MATLAB) can be misleading. While scripting languages generally have the advantage of supporting more compact code, powerful libraries, shorter interactive development cycle and interactive graphics and statistics, compiled languages are usually much faster and, in some sense, offer more freedom. For complex models a hybrid implementation is a sensible option, thereby making use of the advantages of both concepts. For example, for the current implementation of PCLake in the R environment, the model equations are not actually translated into R, instead they are solved in C++ (cf. Soetaert et al., 2010). To do so, R compiles the in C++ coded model equations into a .DLL and invokes this .DLL to numerically integrate the model. Note that while both the OSIRIS and the R implementation use C++ code, this is not exactly the same code because each framework has its own exact specification of the function call to the C++ routine with the ecological process formulations of PCLake. So, while the DATM translators for OSIRIS and R have much in common, there are subtle differences to meet the exact requirements of each framework.

4. Discussion

The DATM approach we here present allows *ecology* to take precedence over *informatics*. We achieve this by formulating the

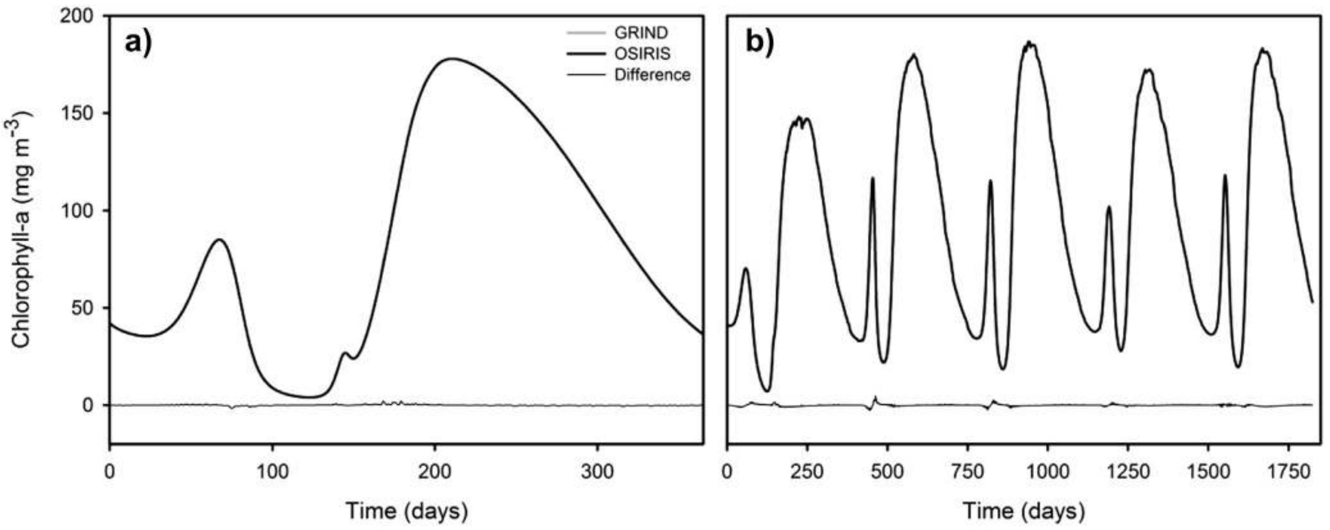

Fig. 2. PCLake benchmark simulation output for chlorophyll-a produced by two different frameworks (OSIRIS and GRIND for MATLAB resp.), for a 'simple' 1-year simulation (a) and a 'complex' multi-year simulation (b) whereby the system is exposed to time series of meteorological forcing, hydrological forcing and transport of matter (e.g. nutrient loading). Also the difference between the simulations is plotted, showing that the output series of the two frameworks are almost identical.

model in the fundamental and universal language of mathematics, and by systematically complementing this mathematical notation with the necessary metadata. The translators create a seamless bridge between the mathematical formulation of the model in the database and the framework-specific implementations.

Experience gained during years of development of PCLake was the main driver behind the development of DATM. PCLake was initially developed in the ACSL framework (Mitchell and Gauthier, 1976), which served as an excellent platform for model development, but where license costs limited the distribution of the model. As this distribution-bottleneck hindered wider use of the model, version 4.08 of PCLake was translated to DUFLOW, a framework that also allows spatial configurations of the model (Jeuken et al., 1999). To further respond to user needs, this version was then translated into DELWAQ and OSIRIS (Mooij et al., 2010). Each translation involved first distinguishing model- from framework-code, and then translating the framework code. Although these translations were semi-automated, each translation represented a big time investment, in which only a few scientists, undaunted by the complexity of the model and specifics of the different frameworks, could effectively carry out the translations and verifications. These efforts monopolized energy away from further model application, analysis and development. The universal mathematical notation we here advocate greatly simplifies the translation process, and makes it much more dynamic and robust at the same time. This allows for direct translation of a new model version in the framework of choice, thereby greatly facilitating the process of model development. Typically, the time needed to develop and test a new translator varies between a few hours for a simple model like the Lotka–Volterra equations to a week for a complex model like PCLake for any given framework.

The erstwhile barriers to framework-switching have led to each framework developing more complex modules to accommodate the growing scope of simulation models. These developments not only make the underlying ecological processes and assumptions more difficult to access, but also require the user to select more options and provide more detail. These developments can in turn reduce the in-depth understanding of the model. Paradoxically, this form of model-framework co-evolution leads to a necessary simplification of a model to make it graspable and useful for ecological theory (Van Nes and Scheffer, 2005; Scheffer and Beets, 1994), whereas the purpose of adding complexity to the framework ought to be to uncover more complex processes in models.

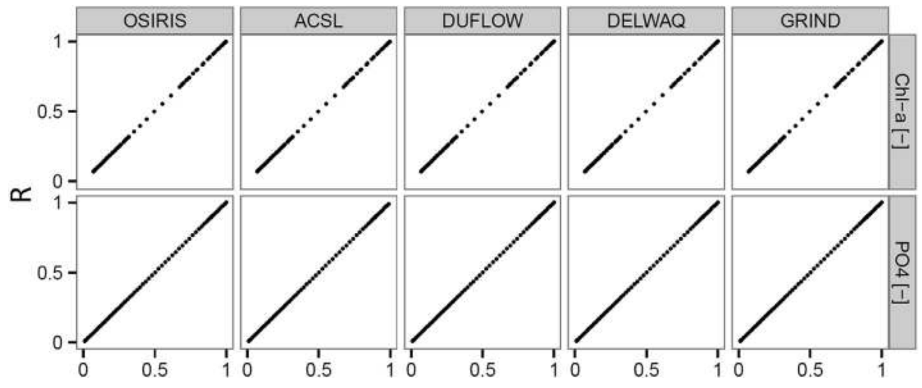

Fig. 3. Illustrative example showing the successful translation of PCLake to different frameworks, whereby the output of the R application is compared with the output of OSIRIS ACSL, DUFLOW, DELWAQ and GRIND for MATLAB respectively, with chlorophyll-a and soluble reactive phosphorus in the pelagic as the dependent variable. Please note that both axis are normalized by dividing each value by the maximum value.

The diversity of analysis tools available across frameworks can greatly enhance our scientific understanding of any given ecological model. In that sense, the database is used to specify where to go, while the different translators and associated frameworks represent ways to get there. One could take route-planning software as a metaphor: the user gives a final destination whereupon the route-planner proposes alternative routes depending on the type of transport one prefers (i.e. bus, train, car, walking, airplane etc.). To explore the ecological code in detail one should go 'by foot', (e.g. using GRIND for MATLAB), while for fast simulation runs an 'airplane' would be more convenient (e.g. OSIRIS). Before entering the territory of spatial complexity of the system with frameworks like DUFLOW and DELWAQ, it might be useful to perform an in-depth analysis of the ecological part of the model in a 0D context. Here, we can exploit the potential of DATM to translate a single code to either a set of ODE's for a general purpose framework of the required ecological component or the PDE's of a spatially explicit water quality modelling framework. To study the asymptotic behaviour of PCLake, translators for bifurcation programs such as MatCont (Dhooge et al., 2003) and AUTO (Doedel et al., 2007) are planned. For the *most* optimal use of the capabilities offered by the different frameworks, however, proper frameworks-specific user knowledge will always be a prerequisite. For the more simple analysis that are provided by most frameworks, however, DATM allows one to stick to the framework one is familiar with and is not forced to learn a new framework.

Experience teaches that DATM also facilitates model simplification by making use of the very existence of a database: providing a clear overview of all model equations and the possibility to label them (e.g. code for spatial dimensioning, hydrology, integration, or user-interface). By means of queries, groups of model equations can easily be identified, grouped and then switched off or simplified. Because columns can be easily duplicated, one can specify multiple versions of the model concurrently in a single table, and then specify which version of the equations is used in a specific simulation. For example, one can easily compare how different types of functional response functions affect model outcome. By "experiments in model structure", DATM is a relatively straightforward tool for assessing model structural uncertainty in addition to input and parameter uncertainty, which is seldom examined (Mooij et al., 2010). DATM thus also potentially allows for model structure optimization, whereby different model structures can be rapidly assessed as part of an optimization process and the most optimal structure is selected (Recknagel et al., 2008). Completing the columns with the necessary meta-information has the additional advantage of contributing to 'good modelling practice' by improving communication among those working with the model (Scholten et al., 2007).

There is increased need for community-based approaches to ecosystem modelling, in order to bring together the knowledge and expertise of ecologists across fields and methodological approaches (Mooij et al., 2010; Trolle et al., 2012). The DATM approach we present here is ideal for building community based approaches: indeed, using a common language (mathematics) and grammar (DATM + translation platform) makes the cross-pollination of ideas and expertise between frameworks, institutes, disciplines and approaches both easier and more attractive. This is not restricted to the field of aquatic ecosystem modelling, as other scientific disciplines can also benefit from a standardized and easily understandable formulation of processes and equations (Jeltsch et al., 2013), allowing one to explore more complex questions in a multidisciplinary setting, and enhancing the interaction with environmental management (Scholten et al., 2007). Additionally, the structure provided allows for easy reuse of pieces of code and processes, thereby preventing 'reinventions of the wheel' (Mooij et al., 2010). To further promote model development, we strongly encourage DATM initiatives to be released under the GNU General Public License (http://www.gnu.org/licenses/gpl-3.0.txt), or the GNU Lesser General Public License (http://www.gnu.org/licenses/lgpl-3.0.txt) so that open sharing of common versions of models is guaranteed.

Emphasis on the model rather than on the framework has an added educational value: teachers can focus on the ecological principles of interest and students can rely on their existing mathematical knowledge to access these principles instead of being first subjected to an often superficial crash-course in a framework's implementation specifics. Our approach thus also makes the model more directly manipulatable by students, irrespective of their framework experiences, and ensures their understanding of model dynamics is based on the ecological model, rather than confounded by framework options. In fact, the Lotka–Volterra DATM example that we presented here and provide as a digital appendix can be of direct use in an educational context.

It is necessary to store the equations in the correct order in the database. With this we mean that each variable must be assigned a value before it is used in the assignment of another variable (in other words, it must first be used as a left hand term before it is used as a right hand term). Some frameworks such as GRIND for MATLAB and ASCL do this sorting automatically, but others do not have this facility. To stay compliant with the latter frameworks, the statements should be ordered already in the database. Fortunately, most compilers or interpreters do provide the user with warning messages accompanied by helpful information when the sequence is violated. Yet, one of the disadvantages of code generators (and other top-level structures which hide implementation details) is that they can make debugging difficult. This is remedied by an iterative procedure, where the user edits and tests the generated code temporarily and then goes back to the table, which gives just another argument for readable code and proper indentation.

We do not claim that our approach is unique in all respects. For instance, both the ECOBAS (http://www.ecobas.org/ecobas/index.html) and SED-ML (http://sed-ml.org/) initiative aim at creating unity in dynamical modelling. ECOBAS provides an overview of ecological models with their metadata and references to the models themselves. SED-ML provides a unifying language for the implementation of dynamical models. DATM balances between those approaches by providing the actual models, but with a focus on the mathematics of the model instead of the informatics. The idea to implement the complete model in a database resembles the design concept of the modelling framework SMART (Kramer and Scholten, 2001). The current version of SMART, however, does not allow translating and exporting models to other frameworks, whereas this is a key-feature of DATM. Automated code translators are already in use at the level of individual frameworks (e.g. SMILE, Muetzelfeldt and Massheder, 2003), although mostly for simpler models. Moreover, there are important advances in establishing a community-based framework for aquatic ecosystem models aiming at unity at the framework level, i.e. the Framework for Aquatic Biogeochemical Models (FABM) (Trolle et al., 2012). A number of the advantages mentioned here are also covered by FABM, such as easy inclusion of new variables and equations, and automatically incorporating different physical assumptions in 0D-3D. DATM complements such efforts – i.e., DATM may also translate models into the FABM framework – thereby providing unique abilities to address some of the challenges and opportunities that remain in the field of aquatic ecosystem modelling (Mooij et al., 2010).

At the onset of this project, our humble aim was to maintain long-term availability and use of PCLake and PCDitch. Happily, this work produced a remarkable and unexpected spin-off: with DATM we have acquired the ability to interactively use multiple frameworks in a single study and even within a single analysis. This dynamic shift in framework use, and more importantly in ecological simulation model analyses, will likely represent a cornerstone in the further development of ecological modelling. As illustrated with the Lotka–Volterra model and the use of Excel and VBA, the ingredients need not be exotic for the pudding to be tasty.

Acknowledgements

The development of DATM for PCLake and PCDitch was financed by the Netherlands Foundation for Applied Water Research (STOWA) project no. 443237, the China-Netherlands Joint Scientific Thematic Research Programme (JSTP) of the Netherlands Organisation for Scientific Research (NWO) project no. 842.00.009, the Netherlands Environmental Assessment Agency (PBL) and the Netherlands Institute of Ecology (NIOO-KNAW). This is manuscript 5609 of the Netherlands Institute of Ecology (NIOO-KNAW).

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.envsoft.2014.04.004.

References

Clemmens, A., Holly Jr., F., Schuurmans, W., 1993. Description and evaluation of program: duflow. J. Irrig. Drain. Eng. 119, 724–734.

Delft Hydraulics, 1995. DELWAQ Version 4.0 Technical Reference Manual. Delft, The Netherlands.

Deltares, 2013. D-Water Quality, Water Quality and Aquatic Ecology Modelling Suite. User Manual – Water Quality and Aquatic Ecology, and Technical Reference Manual – Processes Library Description. Deltares, Delft, The Netherlands.

Dhooge, A., Govaerts, W., Kuznetsov, Y.A., 2003. MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. ACM Trans. Math. Softw. (TOMS) 29, 141–164.

Doedel, E., Paffenroth, R., Champneys, A., Fairgrieve, T., Kuznetsov, Y.A., Oldeman, B., Sandstede, B., Wang, X., 2007. AUTO-07P: Continuation and Bifurcation Software for Ordinary Differential Equations. Available for download from. http://indy.cs.concordia.ca/auto.

Gal, G., Parparov, A., Wagner, U., Rozenberg, T., 2004. Testing the impact of management scenarios on water quality using an ecosystem model. In: Pahl-Wostl, C., Schmidt, C., Rizzoli, C., Jakeman, C. (Eds.), Complexity and Integrated Resources Management, Transactions of the Second Biennial Meeting of the International Environmental Modelling and Software Society, iEMSs: Manno, Switzerland. University of Osnabrück, Germany, p. 88.

Hipsey, M., Romero, J., Antenucci, J., Hamilton, D., 2007. Science Manual: Computational Aquatic Ecosystem Dynamics Model: CAEDYM v3. Centre for Water Research, University of Western Australia, Australia.

Janse, J.H., De Senerpont Domis, L.N., Scheffer, M., Lijklema, L., Van Liere, L., Klinge, M., Mooij, W.M., 2008. Critical phosphorus loading of different types of shallow lakes and the consequences for management estimated with the ecosystem model PCLake. Limnol. Ecol. Manag. Inland Waters 38, 203–219.

Janse, J.H., 1998. A model of ditch vegetation in relation to eutrophication. Water Sci. Technol. 37, 139–149.

Janse, J.H., Scheffer, M., Lijklema, L., Van Liere, L., Sloot, J.S., Mooij, W.M., 2010. Estimating the critical phosphorus loading of shallow lakes with the ecosystem model PCLake: sensitivity, calibration and uncertainty. Ecol. Model. 221, 654–665.

Jeltsch, F., Blaum, N., Brose, U., Chipperfield, J.D., Clough, Y., Farwig, N., Geissler, K., Graham, C.H., Grimm, V., Hickler, T., 2013. How can we bring together empiricists and modellers in functional biodiversity research? Basic Appl. Ecol. 14, 93–101.

Jeuken, M.H.J.L., Janse, J.H., Aldenberg, T., 1999. PCLake, in Anonymous Procesbeschrijvingen DUFLOW, Versie 3 (Chapter 13). STOWA-rapport 99–21.

Kramer, M.R., Scholten, H., 2001. The Smart approach to modelling and simulation. In: Heemink, A.W., Dekker, L., Arons, H.D.S., Smit, I., van Stijn, T.L. (Eds.), Proceedings of Eurosim 2001, Shaping Future with Simulation: 4th International Eurosim Congres, Delft, 2001. TU Delft, The Netherlands.

Leavesley, G., Markstrom, S., Restrepo, P., Viger, R., 2002. A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling. Hydrol. Process 16, 173–187.

Lotka, A.J., 1920. Undamped oscillations derived from the law of mass action. J. Am. Chem. Soc. 42, 1595–1599.

Mitchell, E.E., Gauthier, J.S., 1976. Advanced continuous simulation language (ACSL). Simulation 26, 72–78.

Mooij, W.M., Boersma, M., 1996. An object-oriented simulation framework for individual-based simulations (OSIRIS): *Daphnia* population dynamics as an example. Ecol. Model. 93, 139–153.

Mooij, W.M., Trolle, D., Jeppesen, E., Arhonditsis, G., Belolipetsky, P.V., Chitamwebwa, D.B.R., Degermendzhy, A.G., DeAngelis, D.L., De Senerpont Domis, L.N., Downing, A.S., Elliott, J.A., Fragoso Jr., C.R., Gaedke, U., Genova, S.N., Gulati, R.D., Hakanson, L., Hamilton, D.P., Hipsey, M.R., 't Hoen, J., Huelsmann, J., Los, F.H., Makler-Pick, V., Petzoldt, T., Prokopkin, I.G., Rinke, K., Schep, S.A., Tominaga, K., Van Dam, A.A., Van Nes, E.H., Wells, S.A., Janse, J.H., 2010. Challenges and opportunities for integrating lake ecosystem modelling approaches. Aquat. Ecol. 44, 633–667.

Muetzelfeldt, R., Massheder, J., 2003. The Simile visual modelling environment. Eur. J. Agron. 18, 345–358.

R Development Core Team, 2008. R: a Language and Environment for Statistical Computing. R Foundation for Statistical Computing Vienna, Austria.

Recknagel, F., Cetin, L., Zhang, B., 2008. Process-based simulation library SALMO-OO for lake ecosystems. Part 1: object-oriented implementation and validation. Ecol. Inform. 3, 170–180.

Reichert, P., 1994. AQUASIM – a tool for simulation and data analysis of aquatic systems. Water Sci. Technol. 30.

Scheffer, M., Beets, J., 1994. Ecological models and the pitfalls of causality. Hydrobiologia 275, 115–124.

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. Model. Softw. 22, 743–759.

Soetaert, K., Petzoldt, T., Setzer, R.W., 2010. Solving differential equations in R: package deSolve. J. Stat. Softw. 33 (9), 1–25.

Spaans, W., Booij, N., Praagman, N., Norman, R., Lander, J., 1989. DUFLOW: a Micro-Computer Package for the Simulation of One-Dimensional Unsteady Flow in Open Channel Systems. Bureau SAMWAT, The Hague, The Netherlands.

Trolle, D., Hamilton, D.P., Hipsey, M.R., Bolding, K., Bruggeman, J., Mooij, W.M., Janse, J.H., Nielsen, A., Jeppesen, E., Elliott, J.A., Makler-Pick, V., Petzoldt, T., Rinke, K., Flindt, M.R., Arhonditsis, G.B., Gal, G., Bjerring, R., Tominaga, K., 't Hoen, J., Downing, A.S., Marques, D.M., Fragoso Jr., C.R., Sondergaard, M., Hanson, P.C., 2012. A community-based framework for aquatic ecosystem models. Hydrobiologia 683, 25–34.

Van Liere, L., Janse, J.H., Arts, G.H., 2007. Setting critical nutrient values for ditches using the eutrophication model PCDitch. Aquat. Ecol. 41, 443–449.

Van Nes, E.H., Scheffer, M., 2005. A strategy to improve the contribution of complex simulation models to ecological theory. Ecol. Model. 185, 153–164.

Volterra, V., 1931. Variations and fluctuations of the number of individuals in animal species living together. In: Chapman, R.N. (Ed.), Animal Ecology. McGraw-Hill, New York, USA.

Volterra, V., 1926. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. Mem. Acad. Lincei Roma 2, 31–113.