Extend training material for SPICOSA

provided by

Joachim Maes

SPICOSA WP8

VITO

contact: joachim.maes@vito.be

Plankton dynamics in a bottle.

Introduction

This manual introduces a simple example from the field of experimental ecology to show how an Extend model is developed an how it can be packed as an Extend model block that resides in a model block library. This is indeed an ultimate goal of the SPICOSA project. Here, a step by step guide is presented which subsequently shows how

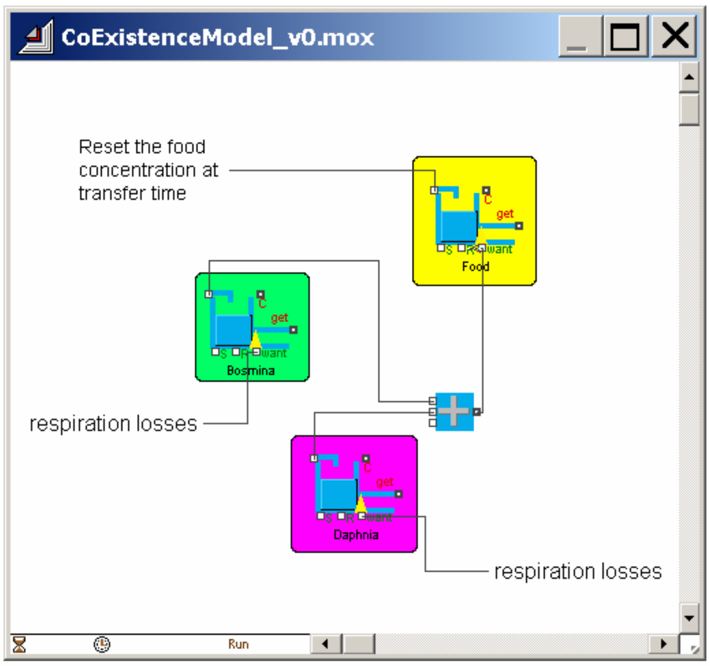
- to construct a model based on a conceptual description
- to calibrate this model using an experimental dataset based on laboratory observations
- to encapsulate the model in a more hierarchical structure
- to encode the model using ModL, the programming language of Extend
- to create a library in which the blocks reside
- to add documentation to the model block so that potential users can use the block

Example model: the coexistence of two cladoceran species

This example is taken from a paper by Soetaert et al[1]. (2002). The simple model describes the competitive interaction between two zooplankton species, Daphnia galeata and Bosmina longirostris, grazing on the same food source (Soetaert et al. 2002). Both species have different maximum growth rates and different half-saturation constants, such that Daphnia is the superior competitor (higher net growth) at high food concentrations, whereas Bosmina achieves higher growth at low food concentrations. They are kept in a culture where they are transferred to new medium with a known amount of food at regular time intervals. The experimental conditions are such that there is no growth of the food in the medium. The model can be used to find out under which conditions of transfer regime and food concentration, both species may coexist or which of the two species will out compete the other species.

A conceptual model of the coexistence of the two plankton species was made in Extend (Fig. 1). The food in the medium and the biomass of both Daphnia and Bosmina represent the stocks of the model (the state variables or the holding tanks). The unit is biomass expressed as mg C per litre. Flows between the stocks increase or decrease the stock level. Each time at transfer time; the contents of the medium is reset to the intial conditions. The stock of food decreases due to grazing. The latter process increases the biomass of the two plankton species while respiration is responsible for stock decrease.

[1] Soetaert K, deClippele V, Herman P (2002) FEMME, a flexible environment for mathematically modelling the environment. Ecological Modelling 151 (2-3): 177-193.



Model parameters

Grazing
AssDaphnia = 0.75
AssBosmina = 0.92
MaxIngDaphnia = 0.27 h^{-1}
MaxIngBosmina = 0.083 h^{-1}
ksDaphnia = 0.98 mgC L^{-1}
ksBosmina = 0.18 mgC L^{-1}

Losses
RespDaphnia = 0.005 h^{-1}
RespBosmina = 0.005 h^{-1}

Transfer
TransferTime = 20 h

Initial conditions
DaphniaT0 = 0.5 mgC L^{-1}
BosminaT0 = 0.5 mgC L^{-1}
FoodInMedium = 0.1 mgC L^{-1}

Fig. 1. Left. Conceptual model of the coexistence of two zooplankton species feeding on the same resource (CoExistenceModel_v0.mox). The yellow box represents the dynamics of the food source. At transfer time, the food concentration is reset at initial conditions. Two species, Bosmina and Daphnia, are consuming food. This process decreases the food stock while it increases the biomass of the two plankton. Biomass is reduced because of losses due to respiration. **Right**. Model parameters used in the nominal model.

The model equations, expressing the rate of change in time, are simple:

(1) $\text{dDaphnia} / \text{dt} = \text{IngestionDaphnia} \cdot\times \text{AssDaphnia} - \text{RespirationDaphnia}$

(2) $\text{dBosmina} / \text{dt} = \text{IngestionBosmina} \cdot\times \text{AssBosmina} - \text{RespirationBosmina}$

(3) $\text{dFood} / \text{dt} = - \text{IngestionDaphnia} - \text{IngestionBosmina}$

At transfer time: Food = FoodInMedium

The ingestion of food by the zooplankton species is limited by the food concentration according to a hyperbolic function and respiration is simply first-order with respect to animal biomass. A fixed fraction (the assimilation fraction) of ingested material is converted to animal biomass.

For Daphnia, respiration and ingestion are calculated as:

$\text{IngestionDaphnia} = \text{MaxIngDaphnia} \cdot\times \text{Food} / [\text{Food} + \text{ks}] \times\cdot \text{Daphnia}$

$\text{RespirationDaphnia}=\text{respDaphnia} \cdot \text{DAPHNIA}$

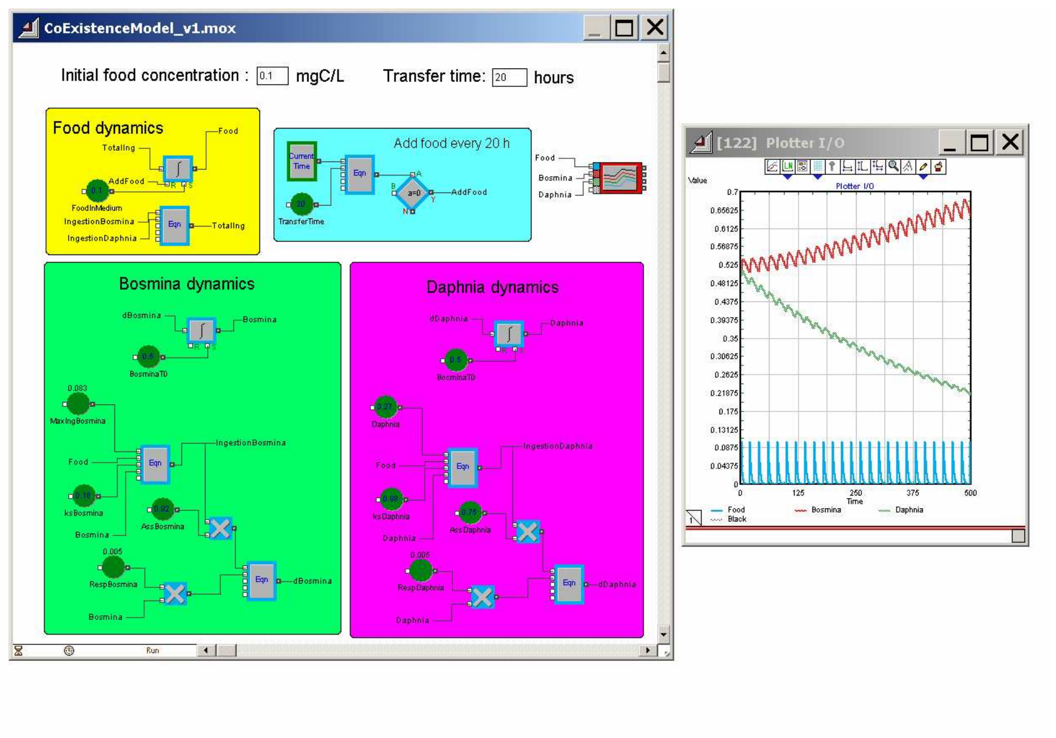
Step 1. First implementation of the model in Extend

The different equations with appropriate parameters (see Fig. 1 for the parameters used in the model) were implemented in Extend. The model is saved as CoExistenceModel_v1.mox and can be downloaded from the internal or the public website. Fig. 2 presents a screenshot of the complete model. This is how the model looks like when you implement the stocks and flows for the first time. For better interpretation, coloured boxes were drawn grouping the stocks and flows of the different state variables of the model. These boxes are nothing more than coloured rectangles. They do not represent any hierarchy in this model. It is important to say that the integration block was used in the simulation, rather than the holding tank. You can use both blocks to simulate the changes of the state variables of the model[2].

Five groups can be distinguished on the model pane. The yellow block assembles the processes that increase or decrease the food concentration. Every 20 hours, the food concentration is reset at 0.1 mgC L^{-1}. This operation is performed using the S and R connectors of the integrator block. Input to the R connector is provided by the decision structure grouped in the blue box. The input of the integrator block is connected to the sum of food ingestion by the two zooplankton species (TotalIng). The green box groups all the processes that affect the Bosmina biomass. The in connector of the integrator receives the product of assimilation and ingestion minus the respiration. The S connector receives the initial concentration of Bosmina in the culture. Ingestion is calculated using the maximum ingestion rate, the food concentration, the half saturation constant and the Bosmina biomass. Respiration is calculated using a respiration rate and the Bosmina biomass. The purple box is a copy of the green box but with parameters for Daphnia in stead of Bosmina. Couplings between the three state variables provide for feed backs. For each of the integrator blocks, the option "Euler (forward)" was selected. This is an appropriate numerical scheme to solve the coupled differential equations. Again, it is perfectly fine to use the holding tank block in stead of the integrator block. Then you need to connect the sources (processes that increase the amount of biomass) to the In connector of the holding tank and the sinks (processes that decrease the amount of biomass) to the Want connector. Finally, in the upper right corner of the model pane the I/O plotter icon presents the output.

[2] There are some differences between the intergrator block and the holding tank block.

If you run the model with a simulation time of 500 hours, you will notice that Bosmina prevails under the present food conditions (0.1 mgC L^{-1} and transfer time of 20 hours, Fig. 2). On top of the pane, a small user dialog was added in order to facilitate the simulation of different food conditions. The two parameters of interest, transfer time and initial food concentration, were cloned using the cloning tool. Manually changing these parameters in the dialog will result in changed values in their respective constant blocks.


CoExistenceModel_v1.mox

Initial food concentration : 0.1 mgC/L   Transfer time: 20 hours

Fig. 2. Left. Implementation of the coexistence model in Extend. **Right.** Under the nominal model conditions, Bosmina (red line) prevails while Daphnia (green line) will be driven to extirpation. The blue line represents the food concentration in the medium which is refreshed every 20 hours.

This model needs a number of improvements. First, it needs **calibration** against a set of experimental data. Most of the parameters used in the model are based on experiments since these two species are heavily studied by experimental ecologists and evolutionary biologist. Notice however, that the respiration rate is for both species equal to 0.005 h^{-1}. For this parameter, no species specific data was available so we can now fit the model to observations by altering these two parameters. This will be explained in the next paragraph.

A next improvement to this model is to **remove duplication**. The dynamics for both plankton species are mathematically the same. Only other parameters are used. The model building blocks that constitute the plankton dynamics make excellent candidates to be grouped in a hierarchical structure or in a new model building block. This would greatly facilitate extending the model by adding a for instance a third competitor. The different options that are provided by Extend to add hierarchy to models will be explored.

A final improvement relates to **proper documentation**. For an ecological modeller who is only interested in simulating the dynamics of competition between these two species for his own purposes, this model is finished after calibration or validation. However, as soon as he needs to share the model with colleagues or if he digs up the model one year later, he needs to invest precious time in finding out how the model works. Why did he use certain parameters, in which publications did he found these parameters, where on his PC are the data are stored to calibrate the model, and so on. Also, colleagues that are interested in elaborating this model by adding different species of plankton and food or by adding different ecological processes that affect the stocks of biomass such as reproduction and mortality will quickly experience that the number of blocks on the model pane will soon become too large. It will become difficult to locate constant blocks or state variables or to follow flows between the stocks. Models that are distributed without proper documentation are simply worthless. Extend offers a suite of possibilities to document the model. Key is to add the information as text to the model blocks themselves, not to store information in separate documents.

Step2: Calibration of the CoExistenceModel using the evolutionary optimizer

The next step shows you how to calibrate this model. In this example, the respiration rate for both species was assumed to be 0.005 h^{-1}, while the other rates differed. In the example we will calibrate the model by changing these two parameters such that the model fits to observations made during the experiment. The example can be consulted by opening CoExistenceModel_v2.mox (Fig. 3). Two more boxes were added to the model pane. The red coloured box groups the experimental data. By clicking on the input data model block you can use the option *Plot Data.* The data describes the biomass of each species at transfer time (20 hours). In correspondence to the model results, the population biomass of Bosmina increases under the nominal food conditions while the population biomass of Daphnia decreases. If you run the model once, two plotters pop up. They both compare the model results with the data. The model seems to overestimate the population biomass of Daphnia while the dynamics of Bosmina match the observations better.

Next, the model will be calibrated using the Evolutionary Optimizer block that resides in the Generic Library of Extend. We will not explain here how evolutionary optimisation works. Briefly, it is a method that uses a genetic algorithm to find an optimal solution given a set of parameters. To start an optimisation process in Extend, you need to place an Evolutionary Optimizer block on the model worksheet.

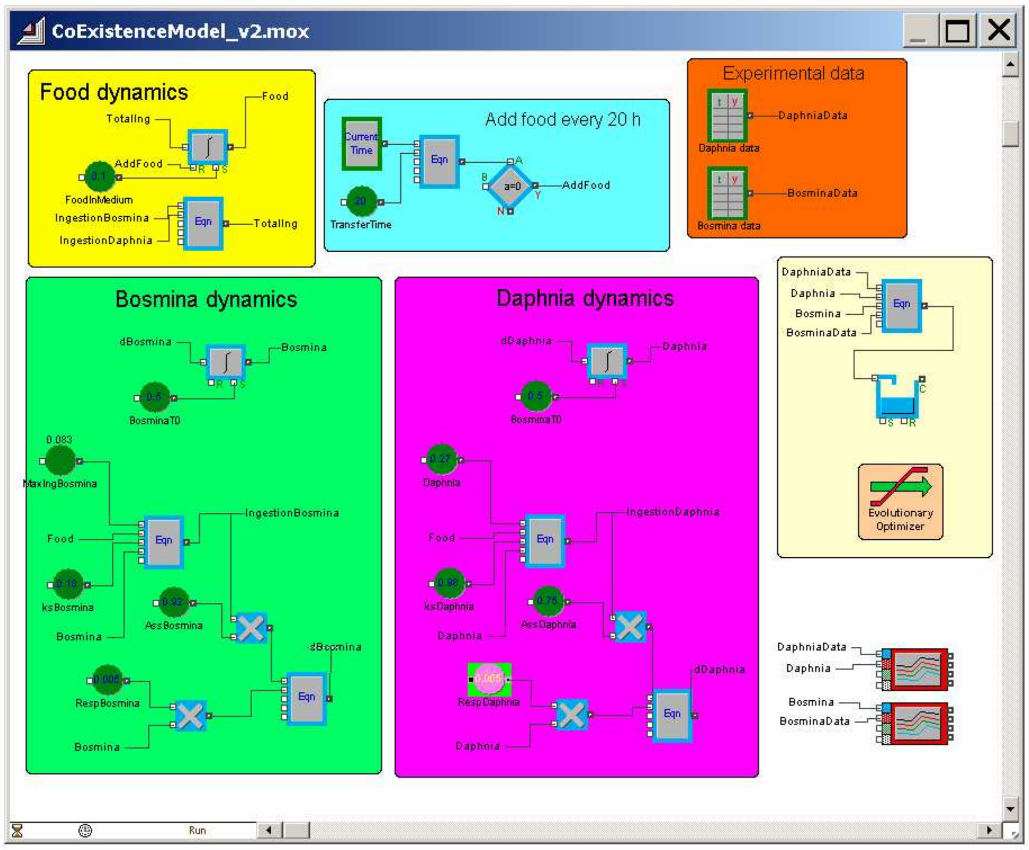


Fig. 3. Screenshot of the coexistence model with on the right hand side of the model worksheet an evolutionary optimizer used for model calibration.

Setting up the objective function in Extend

The next step involves determining which variables in the model need to be optimized. In most cases, optimization means to minimize a cost or the maximize a profit. In this model, we will minimize the cost using a commonly used objective function: the least sum of the squared difference between the observations (the experimental data) and the model outcome. This sum is calculated using an Equation box and the Accumulate block. The objective function thus becomes:

$$\text{MinCost} = \text{sum}\,[(\text{DaphniaData} - \text{DaphniaModelresults})^2 + (\text{BosminaData} - \text{BosminaModelresults})^2]$$

Now, the evolutionary optimizer will try to finetune the respiration rate parameters so that the model matches as good as possible the data.

Setting up the Evolutionary optimizer

In order for the optimizer to calculate this cost equation, it has to have access to the accumulation block in the model. This is done by dragging a clones of the desired variable onto the icon of the Optimizer block. The procedure is:

- Double click on the Accumulation block that contains the variable parameter that you want to use in your cost equation. The variable of interest *Display contents*.
- Select the clone tool.
- Drag that variable onto the closed optimizer block on the model worksheet
- The optimizer block will highlight when the cloned variable can be dropped onto it.

If you now double clock on the Evolutionary optimizer block, you will see that the operation added information about the clone to the Variables table in the Set Cost tab.

Next, use the same procedure to clone the two parameters that will be optimized to onto the closed optimizer block. These parameters are RespBosmina and RespDaphnia. Both are assumed to be 0.005 h^{-1}. So double click on the constant block, select the clone tool, drag the *constant value* onto the closed optimizer block.

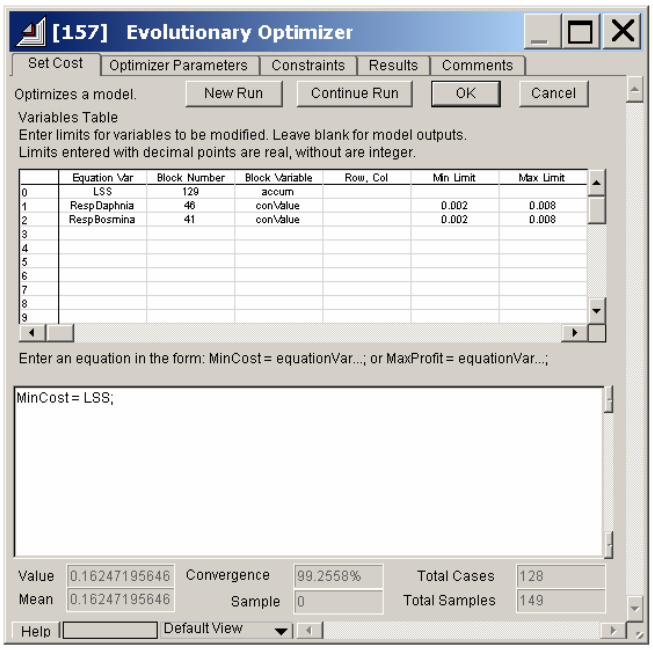
Next, open the evolutionary optimizer block (and if needed, select the Set Cost tab). Now, you can rename the variables and parameters and enter limit values for the two parameters (Fig. 4).

- Change the name of Var0 On the row 0 to LSS. Change the name of Var1 to RespBosmina. Change the name of Var2 to RespDaphnia.
- Enter a minimum limit of 0.002 and maximum limit of 0.008 for the two parameters. During optimization, the generic algorithm will use this range to sample random values. Entering a range will tell to the optimizer that these variables need to be changed in the optimization process.

- Enter the cost function in the textbox: MinCost = LSS. Now, the optimizer will try to minimize LSS which is the least sum of the squared differences between model predictions and data.

- Go to the *Optimizer parameters* tab and click on the *Quicker Defaults, Non-Random Model* button. This quickly sets up all the parameters for a non random model (no random input generator was used), but it limits the number of samples that is taken.

- Click on the *Results* tab to start the optimisation by clinking on *New Run*.



Optimizes a model.

Variables Table
Enter limits for variables to be modified. Leave blank for model outputs.
Limits entered with decimal points are real, without are integer.

	Equation Var	Block Number	Block Variable	Row, Col	Min Limit	Max Limit
0	LSS	129	accum			
1	RespDaphnia	46	conValue		0.002	0.008
2	RespBosmina	41	conValue		0.002	0.008
3						
4						
5						
6						
7						
8						
9						

Enter an equation in the form: MinCost = equationVar...; or MaxProfit = equationVar...;

```
MinCost = LSS;
```

Value	0.16247195646	Convergence	99.2558%	Total Cases	128
Mean	0.16247195646	Sample	0	Total Samples	149

Fig. 4. Optimizer dialog Cost tab for the coexistence model.

After some time, the optimizer comes to a convergence of >95% and the optimization was successful. Note that the original parameter values for respiration rate (0.005) are now replaced by slightly different values so as to minimize the cost function. The respiration rate of Bosmina virtually stayed the same but the respiration rate of Daphnia is now 0.0041 h^{-1}. If you run a new simulation of the model (control + R) you will see that the model now fits the data quite well. We will use these new parameters for subsequent simulations.

This ends the calibration process. Note that another other example of optimisation can be found in the Extend User's Guide on page E238. This example shows how to optimize profits of revenue.

Step 3. Model hierarchy and the reuse of model parts as model building blocks

Adding hierarchy to the model

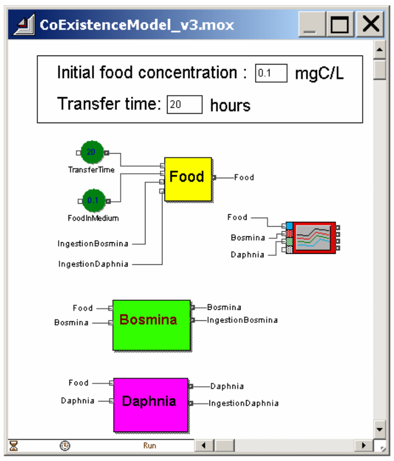
If you take a look again at Fig. 3 , you may notice that the model worksheet is almost full with model blocks, connection lines, data and a plotter. If you would add more blocks, the worksheet becomes larger and you would need the scrollbars to navigate through the model. Clearly, this does not improve the readability of the model. It is time to add hierarchy to the model by encapsulating parts of it into an Extend H-Block or by designing a more generic Extend block. The coloured boxes that group certain model blocks and connection lines suggest already a good candidate for a hierarchical block. As an example, hierarchical blocks were made for each state variable in the model. The result of this exercise is saved as CoExistenceModel_v3.mox (Fig. 5).

When selecting model blocks for inclusion in a H-block, make sure that inputs and outputs into the hierarchical structure are not selected. For instance, the hierarchical food block does not include 4 inputs and 1 output variable.

Also pay attention to the fact that apparently an model output is used as an input. In both the Bosmina and the Daphnia block the output is connected to the input (using the controls "Bosmina" and "Daphnia"). This is possible since the output variable that is the result of calculations during the previous time step is used as input variable during the next time step. The intergrate block of Extend solves numerically the differential equations. Here, the option Forward Euler integration was chosen in the dialog of the Integrate block. The numerical scheme that is used to solve the equation for Bosmina is:

Bosmina(t) = Bosmina(t-1) + timestep $\times$ d[Bosmina(t-1)]/dt

Clearly, the biomass of Bosmina calculated during t-1 is used as input to calculate the biomass of Bosmina during the timestep t.


Initial food concentration : 0.1 mgC/L

Transfer time: 20 hours

Fig. 5. The coexistence model summarized in hierarchical H-blocks.

The result of assembling different blocks into a hierarchical structure is that the model becomes more compact. Any new user can clearly identify tree key players in the model: food, Bosmina and Daphnia. In Fig. 5 the connections or the feedbacks and the inputs between the two plankton species and the food source are not explicitly drawn. In stead, text labels such as Food, Daphnia or IngestionBosmina, were used to provide the connections between the blocks.

Removing duplication

It was already pointed out that there is some duplication in the coexistence model. Essentially, the dynamics of Bosmina and Daphnia are identical. Only other parameters were used. It would therefore be interesting to rewrite the model such that the dynamics of plankton are captured in a single block rather than in two blocks. The single block can then be fed with different parameters, depending on which species you need to model. There is a clear advantage to this procedure. You would save time in constructing a new model for each species that you need to bring in in the model. Further, the new block can be stored in a library and called later by you or by other modellers. Experienced programmers would use a function in Matlab or a subroutine in Fortran. Extend provides two methods for the construction of more generic blocks. Both methods are illustrated here.

A generic H-block for plankton dynamics

Making a sub model such as the Bosmina dynamics more generic corresponds to removing all the features from the model that are specific for Bosmina. Notice that, by collecting the different blocks into the Bosmina hierarchical block, already some specific features were left outside the H-block. The input of food was excluded. The reason for this exclusion is obvious. The food comes from outside the "Bosmina system". It is not generated from within. Changing food conditions alters the food concentration (mgC L^{-1}). This change needs to be communicated to the Bosmina block via the input connector.

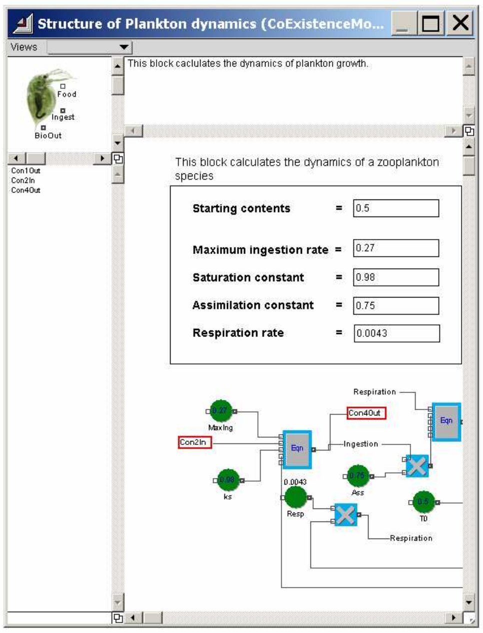
The parameters that are specific for Bosmina are the assimilation constant, the maximum grazing rate, the half saturation constant, the respiration rate and the initial conditions. In fact, these parameters are specific to any plankton species. These parameters will be transferred to a user dialog (or interface). If this step is performed, the Bosmina block will become a zooplankton block with a dialog in which the user can enter species specific parameters. We will store this block in a new library so that it can be used for other models or applications. Unfortunately, Extend does not attach a typical dialog to hierarchical blocks as it does for new blocks. As an alternative, the cloning tool can be used to communicate with the hierarchical zooplankton block.

First, we made a new library by clicking on *New library* in the *Library* menu. The library is saved as CoExistenceModelLibrary and contains the generic H-block 'PlanktonDynamics'.

Next, the Bosmina block was saved in the CoExistenceModelLibrary by Alt + clicking on the Bosmina H-block . This opens the block structure. Now you can save by clicking *Save block to library* from the File menu.

The following step is to change this block. Using google, a picture of plankton species was copied and pasted in the structure pane of the block. Some explanatory text was added. Proper documentation is the focus of the next paragraph, so the text is limited here. Using the cloning tool, we cloned the five parameters of interest into the model pane together with labels to identify the parameters.

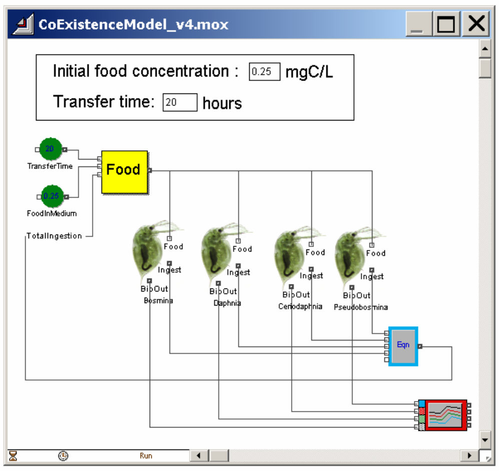
The result of these steps is presented in Fig. 6. The new block resides in the new library. If you drag the block from the library onto the model worksheet and you double click on it, the dialog becomes visible. You can not enter parameters for other species or use the default parameters. Notice that a shortcut was made to connect the biomass output with the biomass input. This connection does not have to made anymore. As a result, the new block counts only one input connector, where the food goes in.



This block caclulates the dynamics of plankton growth.

This block calculates the dynamics of a zooplankton species

Starting contents	=	0.5
Maximum ingestion rate	=	0.27
Saturation constant	=	0.98
Assimilation constant	=	0.75
Respiration rate	=	0.0043



Initial food concentration : 0.25 mgC/L

Transfer time: 20 hours

Fig. 6. Left. Structure window of hierarchical PlanktonDynamics block. **Right**. Application of the new generic H-Block in a new model simulating the coexistence dynamics of four species.

Fig. 6 also shows an application of the new generic block 'PlanktonDynamics' that resides in the library. Now, the coexistence of four species feeding on the same resource is modelled. Each time, we used slightly different parameters using the dialog with cloned parameters. Notice that we also changed the food block so that it can handle inputs from more than two species. You can explore this model in CoExistenceModel_v4 or add species by opneing the CoExistenceModelLibrary and drag more plankton blocks to the model worksheet.

A new block in ModL code to simulate the plankton dynamics

More experienced modellers may wish to implement the equations that describe the dynamics of the plankton in a culture directly in ModL code, the C-like programming language of Extend. Creating a new block using code rather than assembling all the model blocks in a hierarchical H block allows to make optimal use of the dialog features that Extend offers. The Extend developer's reference contains an excellent example to build a new block and in fact, implementing the plankton dynamics into a new block is quite easy.

Firstly, recall that the dynamics of the plankton are solved numerically using forward Euler integration. Let's illustrate Forward Euler integration using the dynamics of Daphnia:

Conceptual model:

(1) Change in biomass = Biomass assimilated by the population – Biomass respired by the population

Mathematical model:

(2) $d[\text{Daphnia}]/dt = \text{AssRate} \times \text{IngRate} \times [\text{Food}/(\text{Food}+\text{ks})] \times [\text{Daphnia}] - \text{RespRate} \times [\text{Daphnia}]$

Numerical solution using the Forward Euler integrator:

(3) $[\text{Daphnia}]^{t+1} = [\text{Daphnia}]^{t} + \Delta t \times d[\text{Daphnia}]/dt$ (where Δt is the time step of the model).

Substituting equation (2) into equation (3) yields a solution for the variable [Daphnia] provided that the start value for the Daphnia biomass $[\text{Daphnia}]^{t=0}$ is given.

The numerical solution of equation (3) can be implemented directly in Extend as ModL code. The parameters of the model can be implemented as dialog items so that they can be changed by a user. The following procedure was used to make a model block simulating the plankton dynamics. A detailed step by step procedure can be found in the Developer's reference on page D41.

- Choose *Build new block* from the *Development* menu. Name the block "PlanktonTank". Open a designated library (here the CoExistenceModelLibrary). Click on *Intall in Selected Library.* To view the structure of the PlanktonTank, click on it while keeping the Alt-key down.

- The dialog window and the structure window open. First, the parameters can be added to the dialog window. This can be done by selecting *New dialog item* from the *Development* menu. Choose *static text* for adding explanatory information or choose *Parameter* to add a user parameter box. In total, five parameters were added: a starting value, three parameters for the feeding process and a parameter for respiration. Some additional text was added to the dialog to provide information.

- Once parameters are named and added to dialog, they appear as Dialog names in the structure window on the left. You do not need to declare these variables as reals, integers or strings in the code.

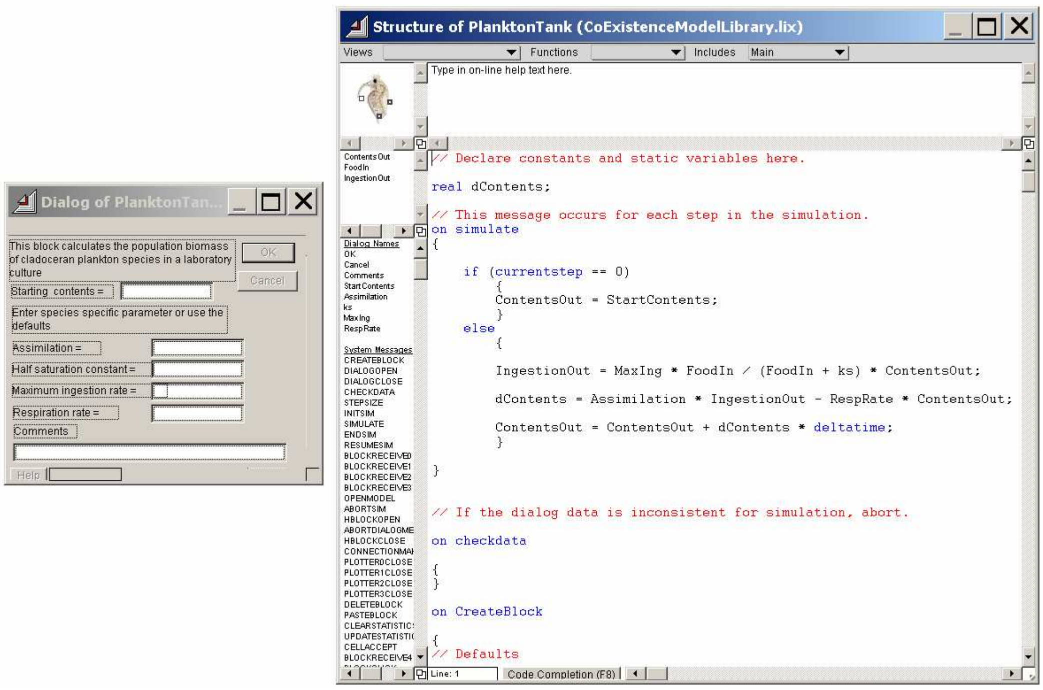
- The default icon is a rectangle, situated in the top left corner of the structure window. You can copy and paste any figure here. Use the Icon tools command on the taksbar of Extend to add inputs and outputs. Names of Inputs should end on "In". Names of Outputs should end on "Out". For instance FoodIn is a valid input name; PlanktonOut is a valid output name. Again, you do not need to declare inputs and outputs in the code if you use the Icon tools button. In our plankton model, three connectors were placed next to a picture of a Bosmina.

- Next, the code is completed. The code for this block is very simple. First, one more variable needs to be declared: `dContents`. Next, the simulation starts by the message `on simulate`. At the start of the simulation (`currentstep = =0`) the biomass of the plankton (`ContensOut`) is equal to the startvalue (StartContents). For all other time steps, the numerical scheme is presented. First, the ingested food is calculated (`IngestionOut`); the next line corresponds to equation (2) while the last line is equation (3). The variable `deltatime` is a global variable in Extend. It is equal to the timestep which can be changed by the user by selecting *Simulation setup* from the *Run* menu.

```
// Declare constants and static variables here.
real dContents;
// This message occurs for each step in the simulation.
on simulate
{
  if (currentstep == 0)
          {
          ContentsOut = StartContents;
          }
  else
          {
          IngestionOut = MaxIng * FoodIn / (FoodIn + ks) * ContentsOut;
          dContents = Assimilation * IngestionOut - RespRate * ContentsOut;
          ContentsOut = ContentsOut + dContents * deltatime;
          }
}
```

- To conclude, we used the message handler "on createblock" to add default values for the parameters.

The resulting dialog and the model structure are pictured in Fig. 7. The block is used in another model simulating the dynamics of three hypothetical species (CoExistenceModel_v5.mox). This model can only be used if the library that is holding the PlanktonTank block (CoExistenceModelLibratry.lix) is installed in de folder Extend6\Libraries.



This block calculates the population biomass of cladoceran plankton species in a laboratory culture

Starting contents =

Enter species specific parameter or use the defaults

Assimilation =

Half saturation constant =

Maximum ingestion rate =

Respiration rate =

Comments

Type in on-line help text here.

```
// Declare constants and static variables here.

real dContents;

// This message occurs for each step in the simulation.
on simulate
{

    if (currentstep == 0)
        {
        ContentsOut = StartContents;
        }
    else
        {

        IngestionOut = MaxIng * FoodIn / (FoodIn + ks) * ContentsOut;

        dContents = Assimilation * IngestionOut - RespRate * ContentsOut;

        ContentsOut = ContentsOut + dContents * deltatime;
        }

}


// If the dialog data is inconsistent for simulation, abort.

on checkdata

{
}

on CreateBlock

{
// Defaults
```

Fig. 7. Structure of the model block PlanktonTank. **Left**. The dialog that was created containing the parameters of the model. **Right**. The model code.

Step 4. Documentation of the model blocks and the model

The newly created Extend block 'PlanktonTank' is now available for any user of Extend who disposes over the library CoExistenceModelLibrary.lix. This library can be put online, it can be downloaded and it can be installed into the folder Extend6\Libraries.

However, as the model block stands now, it still needs documentation. Potential users of this plankton block need to be explained what the block is actually doing. Some users will need to be convinced that the parameters that are used to describe the model are valid and checked. Other users will only be convinced to use this block is they are sure that the model is able to make accurate predictions that are validated against a set of real world observations.

In this paragraph, we will briefly explore the opportunities that Extend provides to document this model block. A standardized Spicosa methodology of documentation of both model blocks will be presented in deliverable D8.4. In this example, we will show how the document the two model blocks that reside in the CoExistenceModelLibrary.lix. We also give hints on how to document a model.

Documentation of the Help function of the Extend blocks PlanktonTank and PlanktonDynamics

If you drag an Extend block onto the model worksheet and you double click on it, the model dialog opens. All Extend blocks whether they are hierarchical H-blocks such as the PlanktonDynamics block or coded blocks such as the PlanktonTank block have a help button in the lower left corner. The help function contains a short description of what the model block does, a description of the dialog items and a description of the connectors. Each block should contain this information so that users are able to implement the block in their models. If you open the help function of the two different plankton blocks, you will notice that they are empty. To illustrate the documentation process, a new library was made (CoExistenceModelLibraryDocu.lix) and the two model blocks were copied to that library. They were given different names as well.

The following procedure is used to document both model blocks

- Open the structure of the model.
- Replace the default help text in the help pane with the documentation.

Note that we added a short introduction and that we explained all the parameters and connectors. We also added names to the connectors using the text tool.

We also used some additional features that come with Extend if you make a model block using the ModL code. If the model block structure window is open, you can add extra information by selecting *Defining new tab* from the *Develop menu*. If you open the dialog of PlanktonTank model block, you will see that the dialog contains two tabs: Model dialog and Concept. The Concept tab describes very briefly the conceptual and mathematical model. In addition the model code itself was updated by using the "//" operator. Application of the double slash in the code window will turn the text into red and can be used for commenting the model. Open the structure of the model and check how the code was commented.