

Abstract

This is the text for the presentation at Skara. See <http://www.dina.kvl.dk/~daisy/> for more information about Daisy. April 16, 2002.

1 Daisy: A Soil–Crop–Atmosphere Model

Daisy: A Soil–Crop–Atmosphere Model

<http://www.dina.kvl.dk/~daisy/>

daisy@dina.kvl.dk

Per Abrahamsen

Laboratory for Agrohydrology and Bioclimatology.

Royal Veterinary and Agricultural University

Denmark

2 About models

- Daisy is a mechanistic, deterministic simulation model.
- Why use models?
 - To understand the system.
 - To make predictions.
 - Decision support / optimization.
 - Save measurements / hidden variables.

Bo originally wanted me to introduce models as a general concept. I hadn't thought about it that way before, so here are my musings.

Both Daisy and Soil_N belong to the same class of models.

mechanistic When we say Daisy is mechanistic, we mean that it attempt to describe the system in terms of the underlying physical, chemical and biological mechanism. The alternative is purely empirical models, which just note the correspondence between measurements, without trying to describe mechanism behind them.

deterministic Daisy always give exact answers, other types of models will give answers in terms of probabilities.

simulation From a know state, Daisy calculate what happens in the future. Diagnostic models will calculate the past, and optimization models will suggest actions to reach a particular goal.

And why do we make models?

To understand Most scientific modelers probably do it because they feel it helps understanding the system.

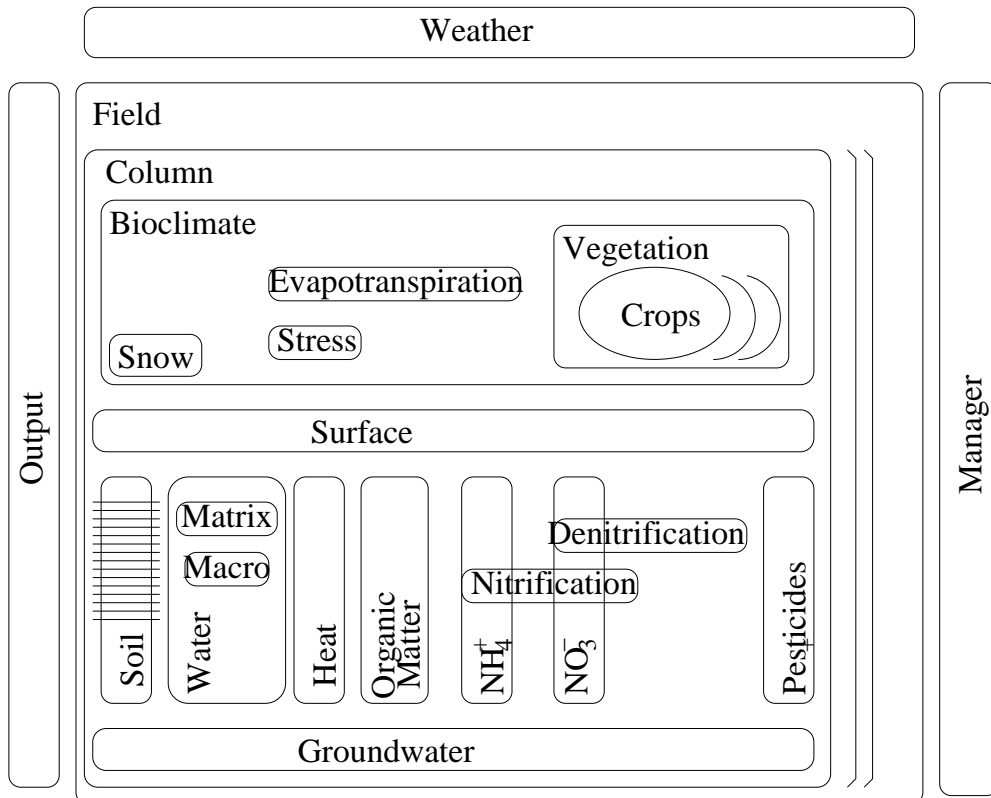
Make predictions What happen if we do this?

Decission support What should be do to achieve this goal?

Save measurements Models can, if we trust them, save us from doing some expensive measurements, and instead let the model calculate them from cheaper measurement.

3 Structure

Daisy



It is possible to visualize a model in a lot of ways, but this is my favorite. It is supposed to bridge the gap between the modeler, the programmer and the

user. Each box here is ideally a separate process in the model, and implemented as a separate module in the program. It also collect related parameters and variables for the user.

Some boxes are within other boxes, which in the model world mean they represent specific subprocess of the more general process og concept. In the program world they are nested submodels.

I'll try to make a fast presentation of the various processes on the picture. The basic concept is the column, a one dimensional representation of a point in the field, which is bordered by the weather at top, and the groundwater at the bottom. We can have many such columns giving a field. At the side, we have a manager who can perform various operations, such as plowing, irrigation and harvesting during the simulation. At the other side we have the output module, which allows the user to log the state of the various processes, also while the simulation run.

- Starting from the top, the weather module provides data about rain, radiation, temperature, and optionally more specialised data such as vapour pressure and wind, at either a daily or hourly timescale.
- The bioclimate distribute the rain and radiation between the soil surface and the vegetation module. The vegetation module further distribute the weather input between the crops on the field.
- Below the surface, we have the soil module containing the physical parameters for the soil, as well as dividing the into a number of numeric layers.
- The water module module provides matrix and macro transportation of the water, using the surface and groundwater as border conditions, and the root uptake as a sink term.
- We also calculate the soil heat, using the air or rain temeprature as an upper border, and the groundwater temperature as a lower border. The heat affects both water transportation, crop emergence, and nitrogen processes.
- The organic matter module keep track of the soil humus and added organic matter, both in the form of crop residuals and fertilizer. This give a mineralization – or immobilization – that feeds the mineral nitrogen in the soil.
- The mineral nitrogen are transported with the water, and mineralization, crop uptake, nitrification and denitrification are all sink or souce terms.
- Finally we can have a number of pesticides, which are transported with the water, and can decompose with a rate that depend on the soil state.

4 Crop and Organic Matter

We can go further into each of the subprocesses here. For example the standard crop model, which is quite complex, and give good production estimates, and the organic matter model. Our organic matter is divided into added matter, biomass, and humus. Each of these are further divided into a fast and a slow pool. This division have made it possible to get good results both on the short range and the long range. As an example of long range, there are at least two projects using Daisy in conjunction with global change. Large amount of carbon is stored in farm land, what will happen to this when temperature and CO₂ rise?

5 Flexibility

When I started 7 years ago, there already was a working implementation of Daisy. I have seen the original implementation, and in my professional opinion, it was rather good code. So why hire a computer scientist and start over?

Because the original code had reached its limits. There were lots of interesting projects who want to use the program, in very different directions. It would be hard, with the original code base, to follow any of the directions, and impossible to follow them all. What was needed was a more flexible code base. And that kind of stuff was what makes a programming task interesting to a computer scientist.

Here is what we did.

- – Replace each process.
 - * Fast vs. precise
 - * Simple vs. complex.
 - * Amount of available information.
 - * Research.
 - * Reuse old parameterizations.
- Use defaults.
- Many columns, many crops.
- Use by other models.

With the new code base, each “box” in this figure can potentially have a number of different implementations, each representing different models of the same process. It is then up to the user to specify which model that is most appropriate for his needs. Some examples are:

- Usually, we spend half the computer time solving Richard's equation in matrix transport. Replacing it with a linked reservoir model can thus cut the simulation time in half. Seen over a longer period the difference in the results even out. However, Such speed optimizations are only worthwhile for very large simulations, usually the simulation take less than a minute per year per column.
- We have a complex, mekanistic crop model that calculate the content of various crop partitions at all times, taking account phtosynthesis, and water and nitrogen stress. It is expensive to paramterise, usually requiring new field experiments for data. We have paramterizations for most common crops, but some users want to make simulations with other crops, but can't affort the experiments. So we have a simpler crop model that can be parameterized from easily available data for such use. The simple model is not useful for production simulation, but acceptable for environmental simulations.
- We have three different models for evapotranspiration, one that just get the data from the weather file, one that uses a simple Makkink to calculate it, and one that uses the FAO recommended Penman-Monteith for the data. Makkink is preferable to Penman-Monteith when the weather data are of low quality.
- People doing research in specific processes can use Daisy as a framework in which to test new models of the process.
- When we implement a new and better model for a certain process, users can still use their parameterizations of the old model, because Daisy can contain both the old and the new model at the same time.
- During the rewrite, we also implemented the ability to have many simulate columns at the same time, and to have multiple crops on the field competing for light and water.
- The multiple columns has also allowed Daisy to be "plugged in" to a groundwater model, Mike/SHE, providing the upper boundary for that model.

6 Column definition

There are a commercial graphical interface for Daisy, and two free graphical interfaces under development. But I think they are all a waste of time. What Daisy is all about is reading a file specifying the setup, and writing a number of files with the results of the simulation. The setup file language is extremely flexible, and you will soon be much more productive with that than any graphical interface, as soon as you have learned to balance paranthese.

Here is an example of a setup. It takes three slides to show.

```

(weather default "weather.dwf")

;; We have some very sandy soil.
(defhorizon Ap default
  (clay 8.0 [])
  (silt 10.5 [])
  (coarse_sand 63.4 [])
  (fine_sand 16.5 [])
  (humus 1.12 [])
  (SOM_fractions 0.66 0.34 []))

(defhorizon C Ap
  (humus 0.12 []) ;Less humus.
  (SOM_fractions 0.80 0.20 [])) ;Slower background minerali

;; We build the column from the horizons.
(defcolumn Andeby default
  "The B.And farm, Andeby, 2002."
  (Soil (horizons (-20 [cm] Ap) (-2.5 [m] C))
    (MaxRootingDepth 60.0 [cm]))
  (Groundwater deep))

;;; ...

(defhorizon B default
  (clay 8.0) (silt 10.5) (coarse_sand 63.4) (fine_sand 16.5)
  (humus 1.12)
  (SOM_fractions 0.70 0.30)
  (hydraulic M_vG
    (K_sat 10 [cm/h])
    (Theta_res 0.05 [cm^3 H2O/cm^3])
    (Theta_sat 0.424 [cm^3 H2O/cm^3])
    (alpha 0.069 [cm^-1])
    (n 1.527)))

```

Here, we first define two horizons, A and C. For each horizon we must

specify the texture, the amount of organic matter, and two magic numbers that divided the organic matter into a fast and a slow pool.

In the C horizon, we use one of the tricks the language allows. We define it to be identical to the A horizon, except for the humus.

Using the already defined horizon, we construct the column. We also need to tell it how deep the roots may reach, and where the groundwater is located. That is all the information we need about the column.

Which is not quite true. Daisy will attempt to guess the hydraulic properties from the texture, i.e. water will run faster through sand than clay, but you should really specify them directly. They are such an essential part of the system, and the pedotransfer functions to guess them are very unreliable.

At the bottom of the slide, I have an example where we use a Mualem – van Genuchten model for the hydraulic parameters. We have a dozen or so models to choose from.

7 Management definition

Here is a management definition.

```
;; Use standard parameterizations.
(input file "tillage.dai")
(input file "crop.dai")

;; Simulation start date.
(time 1986 12 1 1)

(manager activity
  (wait (at 1987 3 20 1))
  (plowing)
  (wait (at 1987 4 4 1))
  (fertilize mineral
    (weight 100.0 [kg N/ha])
    (NH4_fraction 0.5 []))
  (wait (at 1987 4 5 1))
  (progn
    (sow "Grass")
    (sow "Spring Barley"))
  (wait (or (crop_ds_after "Spring Barley" 2.0)
    (at 1987 9 5 1)))
  (harvest "Spring Barley"))
```

```

(wait (at 1987 9 8 1))
(fertilize mineral
  (weight 80.0 [kg N/ha])
  (NH4_fraction 0.5 []))
(wait (at 1987 10 10 1))
(harvest "Grass"
  (stub 8.0 [cm]) ;Leave 8 cm stub.
  (stem 1.00 [])) ;Harvest everything above
(wait (at 1988 4 1 1))
(stop))

```

We have some standard tillage and crop parameterizations in the tillage.dai and crop.dai file, which we read first. Then we set the start time. And then we tell Daisy when the manager can do. The management specification is one area where we have done a lot of work, users seem to have very different needs, so it is a place where the flexible design have paid off. Here we use the “activity” model for management specifications. This models simply perform a number of tasks in sequence. Here we wait for specific time, then perform some action like plowing or harvesting. Finally, we stop the simulation.

8 Rotation

Users typically store parameterizations for horizons and columns in libraries, so they can reuse them in future simulations. For management, we can do that as well. Typically, we describe the management for each crop, and then combine these descriptions in rotations. This saves a lot of work.

```

;; Spring Barley management.
(defaction sbarley activity
  (wait_mm_dd 3 20)
  (plowing)
  (wait_mm_dd 4 10)
  (fertilize pig_slurry (weight 100.0 [T w.w./ha]))
  (wait_mm_dd 4 15)
  (seed_bed_preparation)
  (sow "Spring Barley")
  (wait_mm_dd 7 1)
  (wait (or (crop_ds_after "Spring Barley" 2.0 []) ;Ripe
    (mm_dd 10 1)))
  (harvest "Spring Barley"))

```

```
(defaction rotation_bbgr activity
  sbarley sbarley grass rye)
```

```
(manager activity
  rotation_bbgr rotation_bbgr rotation_bbgr rotation_bbgr rotation_bbgr)
```

9 Output definition

Finally, we specify what to log.

```
;; Create these log files.
(output harvest
  ("N Balance" (when monthly)
    (from 0 [m]) (to -1 [m]))
  ("Crop Production"
    (set "$crop" "Spring Barley")
    (where "sbarley.dlf"))
  (checkpoint (when (at 1987 8 7 6))))
```

We have 47 predefined log models to choose from, some containing information for general balances, and some for specialized submodels. And the user can define more. Here we ask for all harvest information, information to create a nitrogen balance in the first meter of the soil on a monthly basis, information about the development of the spring barley, and a checkpoint.

A checkpoint is file where Daisy save information about the current state, so you later can restart the simulation from that particular point. It has the same format as the setup files, so you can inspect every aspect of the state, and even change stuff before restarting.

One project have used this to “update” the crop state with measured satellite data. I think this could be relevant for precision farming as well.

The output format of the other log models are tab separated column, with a header containing information about how it was generated. The tab separated columns means you can load it into a spreadsheet, such as Excel. We also have a specialized graph viewer, which can even animate soil content data. This is more useful than it sounds, especially for teaching purpose.

10 Applications and Results

And now for some results.

- First, we have two examples of spatial variation. Both contain a large number of measurements within a single field, and we have then ran Daisy to simulate the nitrogen content as various depth. As you can see, there is a large variation. The bars are measured values, the difference here is due to a single large precipitation event, where we believe the matrix was bypassed by macropores. At the time, Daisy did not support macropores, thus the difference. The first slide is a sandy soil, where the variation is greater than in the clay soil on the second slide.
- This shows simulated and measured crop production in a remote sensing project. The data is probably better than for most simulations, since we had very good data available for that project.
- This shows latent heat flux with two different kind of measurements, TDR and eddi-covarians, and two different simulations. One simulation with “normal” climate data, and one using NDVI input. This is local measures, but for the RS model the intent was to use satelite data.
- This show detailed measurement and simulated CO₂ flux. It show how a model can be used to fill the gaps in the measurements, so we can create a carbon balance. It also shows Daisy used for very detailed simulations.
- Here Daisy have been used to simulate N leaching for all of Denmark. The first show (with read) an increase in apprlied fertilization the last 10 years, where the second shows a decrease in leaching in the same period. It demonstrates how farmers have become better in the period, and how Daisy can be used on the large scale as well.

This concludes my presentation.