Jun 17th, 2:00 PM - 3:20 PM

The use of object-oriented programming concepts for documenting a model ensemble

Georgii A. Alexandrov

A.Mm Obukhov Institute of Atmospheric Physics, Russian Academy of Sciences, g.alexandrov@ifaran.ru

International Environmental Modelling and Software Society (iEMSs)
7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA
Daniel P. Ames, Nigel W. T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

The use of object-oriented programming concepts for documenting a model ensemble

Georgii A. Alexandrov [a]

[a] *A.M. Obukhov Institute of Atmospheric Physics, Russian Academy of Sciences, Pyzhevsky 3 Moscow, Russia (g.alexandrov@ifaran.ru)*

Abstract: The spread of cloud computing services gives community modelling a reasonable opportunity to become a reality in scientific research. However, even if models will be deployed in clouds, and model codes will be open for re-use, there will be little progress in community modelling in the lack of consensus standards for model documentation. This paper is to discuss a conceptual framework for developing such standards. The proposed conceptual framework is based on the idea that a new model is often a modification of an old one, and hence, the similarities between the models of the same environmental process could be expressed in terms of object-oriented programming (i.e., as either inheritance or polymorphism). The advantages of such conceptual framework are illustrated with an ensemble of so-called production efficiency models (PEMs) which are used for modelling primary production of terrestrial ecosystems.

Keywords: model ensemble; gross primary production; standards for model documentation

1 INTRODUCTION

Environmental modelling is a field of science that has been highly developed over the past 30-40 years. The number and diversity of models reach the threshold at which the progress in an area of environmental research is achieved through improving performance of a multi-model ensemble, not a single best model (Alexandrov et al., 2011). Therefore, the future of environmental modelling depends more and more on the infrastructure for sharing models and on the ”culture of community modelling”.

The challenge of building community modelling and information sharing culture was thoroughly discussed by Voinov et al. (2008) who said, among other things, ”documentation is a crucial part of the process if we anticipate others will use and take part in the development of our models”. They also pointed out that environmental modelling is an iterative process: a model developed in the course of one research project is often modified in the course of another project. That is, the use of a model suggests its further development. A model that cannot be reproduced and modified is like a theorem without proof: its scientific value is questionable.

Research journals formally requires that results be reproducible. However, the results of computational experiments are more than often reported informally, and are not accompanied by the code which is used to produce them. The problem of computational reproducibility is addressed through development of scientific workflow systems (Freire and Silva, 2012). Although such systems have many advantages, the model description written using standard mathematical notation is much more ”portable”.

Therefore, the ultimate goal of the study reported in this paper is to propose a template for documenting the iterative development of a model ensemble that will be comparable with standard mathematical notation in sense of ”portability” and, at the same time, will be comparable with workflow systems in

sense of computational reproducibility. The feasibility of the approach is illustrated using the case study of production efficiency models (PEMs) of primary production in terrestrial ecosystems.

2 METHODS

Which of the programming languages is more close to the standard mathematical notation in sense of generality and readability? The ideas underlying Python syntax (http://legacy.python.org/dev/peps/pep-0020/) are seemingly close to those of mathematical notation. To understand the pieces of Python code which are used in the template for documenting model ensemble presented in the section 'Results' one need not be an expert in Python: readability is a feature of this language (Perez et al., 2011).

A Python program is divided into a number of logical lines. A comment starts with a hash character and ends at the end of the physical line. The indentation level of the line is used to determine the grouping of statements. An entirely blank logical line terminates a multi-line statement.

For example, the linear function $f(x) = ax + b$ is defined by the following piece of code:

```
def f(x,a,b):
     y=a*x+b
     return y
```

The ensemble of linear functions can be defined using class definitions. For example, if the linear function $f(x) = x + 1$ is defined as the base class:

```
class LinearModelVersion1:
        a=1
        b=1
        def f(self,model):
                y=model.a*self+model.b
                return y
```

then the linear function $f(x) = 2x + 1$ can be defined as the derived class:

```
class LinearModelVersion2(LinearModelVersion1):
        a=2
```

This feature of the class definitions is called 'inheritance'. It provides an efficient way for documenting the differences between the old and the new versions of the model. The derived class LinearModelVersion2 differs from the base class LinearModelVersion1 only by the value of 'a', and so one need not rewrite the definition for the function f(x).

Perhaps, some explanation should be given here for the meaning of variables 'self', 'model', 'model.a', 'model.b'. The variable 'self' is merely an independent variable. One may use 'x' instead of 'self' here. However, there is a tradition to name the first argument in the function defined within a class as 'self'. The variable 'model' has two attributes 'a' and 'b'. Their full names are 'model.a' and 'model.b', respectively. If 'model' is an instance of LinearModelVersion1, then 'model.a'=1 and 'model.b'=1; and if 'model' is an instance of LinearModelVersion2, then 'model.a'=2 and 'model.b'=1.

The snapshot of the Python shell shown at the Figure 1 provides further details regarding the usage of the class definitions. The class definitions saved in the file 'LinearModel.py' are imported in the

first command line. The LinearModelVersion1 is selected in the second command line. The values of model parameters ('a' and 'b') are printed out at the next lines. And the string 'model.f(1,model)' gives the result applying LinearModelVersion1 to the value of independent variable which is equal to 1. The other command lines do the same things for LinearModelVersion2.

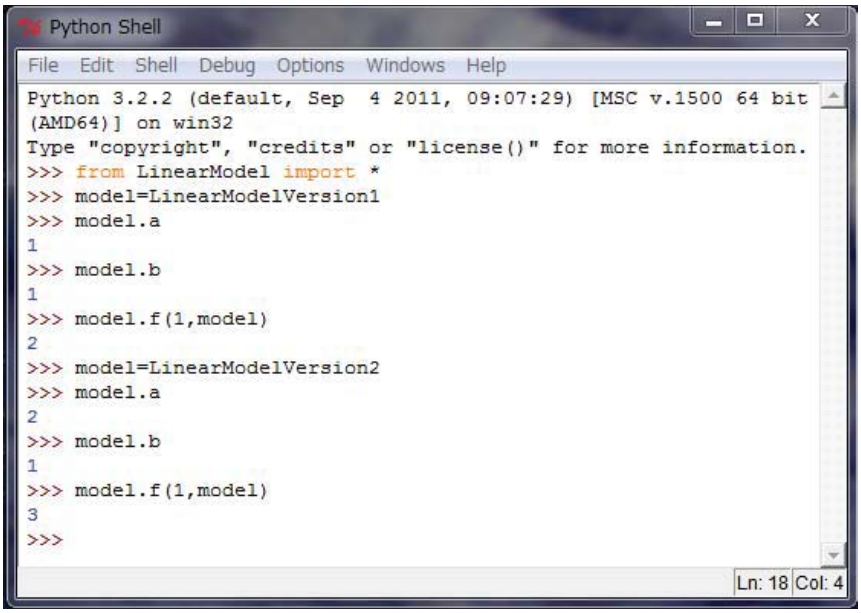

Figure 1. A snapshot of the Python shell illustrating some basic features of Python class objects discussed in the Methods.

It is also important for collaborative development of the model ensemble that Python is provided by public cloud computing services. The best example is SageMathCloud (https://cloud.sagemath.com/), a free service supported by the University of Washington, the National Science Foundation and Google. This service integrates functionality of several mathematical packages (Erocal and Stein, 2010) and makes it accessible through common Python-based interface.

3 RESULTS

The basic concept of production efficiency models (PEMs) suggests that gross primary production (GPP) and net primary production (NPP) of terrestrial ecosystems can be calculated as the product of absorbed photosynthetically active radiation (APAR) and light use efficiency (LUE). The first global NPP model based on APAR used a single value for LUE, 1.25 gC/MJ PAR, and applied it to all biomes (Heimann and Keeling, 1989) and seasons. Since NPP to GPP ratio is often assumed to be equal to 0.5, the basic PEM model for GPP can be written as follows.

```
class PEMver0rev0build0:
        units={'LUE': 'gC/MJ PAR', 'APAR':'MJ/sqm/day'}
        LUE=2.5
        def gpp(self,model): #self stands for APAR
                res=model.LUE*self
                return res
```

The PEMs that differ from the basic PEM only by the value of LUE represent the various 'builds' of the basic PEM. For example, LUE= 1.1 gC/MJ PAR in the C-Fix model (Verstraeten et al., 2006), and so the following class definition is included into the PEM ensemble:

```
class PEMver0rev0build1(PEMver0rev0build0):
            LUE=1.1
```

Ruimy et al. (1994) used a range of ecosystem specific values. Such changes could be named as a 'revision' of the basic model:

```
class PEMver0rev1build0(PEMver0rev0build0):
    LUE={'equatorial and tropical moist forests':0.62,
             'tropical and subtropical dry forests':0.37,
             'temperate deciduous forests': 1.01,
             'temperate and subpolar coniferous forests': 1.57,
             'temperate grasslands': 1.26,
             'deserts': 1.26,
             'tundra and bogs': 1.26,
             'cultivation': 2.07}
    def gpp(self, model): #self stands for APAR
        vegtype=list(model.LUE.keys())
        res={x:model.LUE[x]*self for x in vegtype}
        return res
```

The PEMver0rev1build0 can serve as a base model for various 'builds'. For example, the estimates of LUE derived from Miami NPP model for the same vegetation types (Field et al., 1995) differ from those that were used by Ruimy et al. (1994), and so the following class definition is included into the PEM ensemble:

```
class PEMver0rev1build1(PEMver0rev1build0):
    LUE={'equatorial and tropical moist forests':0.818,
             'tropical and subtropical dry forests':1.094,
             'temperate deciduous forests': 0.9,
             'temperate and subpolar coniferous forests': 0.856,
             'temperate grasslands': 0.994,
             'deserts': 1.526,
             'tundra and bogs': 1.18}
```

Low air temperature, high vapour pressure deficit and other environmental conditions may reduce LUE. The effects of such environmental conditions are taken into account by multiplying the maximum value of LUE (LUEmax) by so-called scalars. For example, the MODIS algorithm for calculating GPP (Running et al., 1999) takes into account the minimum daily air temperature (TMIN) and vapour pressure deficit (VPD) – that is, LUE is calculated as the product of LUEmax, TMINscalar(TMIN), and VPDscalar(VPD). A model that taking into account various environmental conditions differ conceptually from the models discussed above, and therefore could be named as a next 'version' of PEM. Thus, MODIS algorithm for calculating GPP (Running et al., 1999) is included into the model ensemble under the name 'PEMver1rev0build0':

```
class PEMver1rev0build0:
        units={'LUE': 'gC/MJ PAR',
               'TMIN': 'Celsium',
```

```
       'VPD': 'Pa',
       'APAR':'MJ/sqm/day'}
LUEmax={'evergreen needleleaf forest':1.008,
        'evergreen broadleaf forest':1.159,
        'deciduous needleleaf forest':1.103,
        'deciduous broadleaf forest':1.044,
        'mixed forest':1.116,
        'grassy woodland':0.800,
        'wooded grassland':0.768,
        'closed shrubland':0.888,
        'open shrubland':0.774,
        'grasslands':0.680,
        'croplands':0.680
        }
TMINmax={'evergreen needleleaf forest':8.31,
        'evergreen broadleaf forest':9.09,
        'deciduous needleleaf forest':10.44,
        'deciduous broadleaf forest':7.94,
        'mixed forest':8.50,
        'grassy woodland':11.39,
        'wooded grassland':11.39,
        'closed shrubland':8.61,
        'open shrubland':8.80,
        'grasslands':12.02,
        'croplands':12.02
         }
TMINmin={'evergreen needleleaf forest':-8.00,
        'evergreen broadleaf forest':-8.00,
        'deciduous needleleaf forest':-8.00,
        'deciduous broadleaf forest':-8.00,
        'mixed forest':-8.00,
        'grassy woodland':-8.00,
        'wooded grassland':-8.00,
        'closed shrubland':-8.00,
        'open shrubland':-8.00,
        'grasslands':-8.00,
        'croplands':-8.00
         }
VPDmax={'evergreen needleleaf forest':2500,
        'evergreen broadleaf forest':3900,
        'deciduous needleleaf forest':3100,
        'deciduous broadleaf forest':2500,
        'mixed forest':2500,
        'grassy woodland':3100,
        'wooded grassland':3100,
        'closed shrubland':3100,
        'open shrubland':3600,
        'grasslands':3500,
        'croplands':4100
        }
VPDmin={'evergreen needleleaf forest':650,
        'evergreen broadleaf forest':1100,
        'deciduous needleleaf forest':650,
        'deciduous broadleaf forest':650,
        'mixed forest':650,
        'grassy woodland':930,
```

```
            'wooded grassland':650,
            'closed shrubland':650,
            'open shrubland':650,
            'grasslands':650,
            'croplands':650
            }

        def TMINscalar(self,xmin,xmax):
                if self < xmin:
                       res=0
                elif self > xmax:
                        res=1
                else:
                        res=(self-xmin)/(xmax-xmin)

                return res

        def VPDscalar(self,xmin,xmax):
                if self < xmin:
                       res=1
                elif self > xmax:
                        res=0
                else:
                        res=(xmax-self)/(xmax-xmin)

                return res

        def gpp(self, model): #self stands for conditions
                vegtype=list(model.LUEmax.keys())
                res={x:\
                model.LUEmax[x]*self.APAR\
                *model.TMINscalar(self.TMIN, model.TMINmin[x],model.TMINmax[x])\
                *model.VPDscalar(self.VPD, model.VPDmin[x],model.VPDmax[x]) \
                for x in vegtype}
                return res
```

The PEMver1rev0build0 could serve as a base class for various 'builds' and 'revisions' that would cover many of PEMs reviewed by McCallum et al. (2009). The PEMs of this sort need a comprehensive description of environmental conditions. The set of environmental conditions needed by PEMver1rev0build0 is described by the following class definition:

```
class TypicalConditionsVer1rev0build0:
      APAR=15
      TMIN=15
      VPD=1000
```

The snapshot of the Python shell, Figure 2, illustrates the use of PEMver1rev0build0 and TypicalConditionsVer1rev0build0 class definitions.

The class definitions saved in the file 'PEMensemble.py' are imported in the first command line. The PEMver1rev0build0 model is selected in the second command line. The TypicalConditionsVer1rev0build0 set of environmental conditions is selected at the third command line. The estimates of daily GPP, in gC per square meter per day, are calculated for given environmental conditions and the model at the fourth command line.

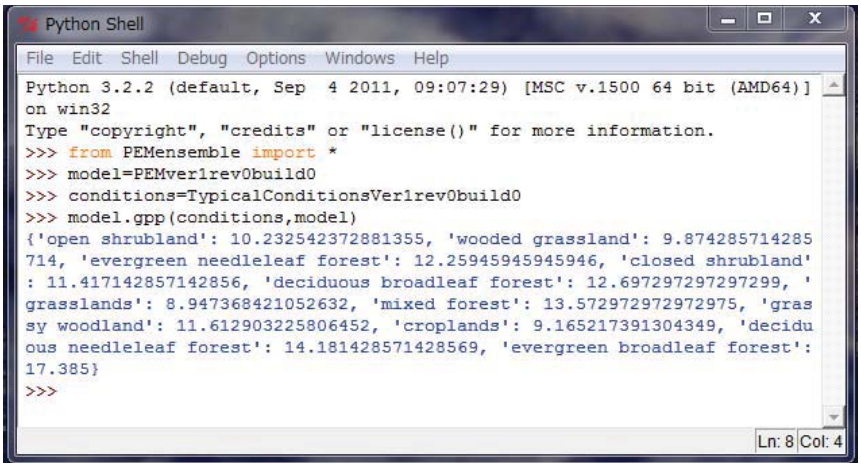

Figure 2. A snapshot of the Python shell illustrating the use of PEMver1rev0build0 and TypicalConditionsVer1rev0build0 class definitions.

4 Conclusions and Recommendations

The models which are used for modelling the same environmental process sometimes differ in details, not in general structure. In such cases a model ensemble may look like a collection of 'builds' and 'revisions' of a base model. The case study of the ensemble of production efficiency models presented in this paper supports the idea that the use of the object-oriented programming concept of inheritance may help in documenting modifications of the base model. It also demonstrates the advantages of Python notations for model documentation. Python notation seemingly provides a good balance between "portability", which is a feature of standard mathematical notation and computational reproducibility, which is a feature of scientific workflow systems. The collection of class definitions presented in this paper provides a starting point for discussing a generic template that can be used for documenting ensembles of environmental models.

Acknowledgments

This research is connected to the project "Seasonal changes in atmospheric CO2 caused by the terrestrial carbon sink " that received financial support from the Russian Foundation for Basic Research, grant No 13-05-00781.

References

Alexandrov, G., Ames, D., Bellocchi, G., Bruen, M., Crout, N., Erechtchoukova, M., Hildebrandt, A., Hoffman, F., Jackisch, C., Khaiter, P., Mannina, G., Matsunaga, T., Purucker, S., Rivington, M., and Samaniego, L. (2011). Technical assessment and evaluation of environmental models and software: Letter to the editor. *Environmental Modelling and Software*, 26(3):328 – 336. Thematic issue on the assessment and evaluation of environmental models and software.

Erocal, B. and Stein, W. (2010). The sage project: Unifying free mathematical software to create a viable alternative to magma, maple, mathematica and matlab. In Fukuda, K., Hoeven, J., Joswig, M., and Takayama, N., editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 12–27. Springer Berlin Heidelberg.

Field, C. B., Randerson, J. T., and Malmstrom, C. M. (1995). Global net primary production: Combining ecology and remote sensing. *Remote Sensing of Environment*, 51(1):74 – 88.

Freire, J. and Silva, C. (2012). Making computations and publications reproducible with vistrails. *Computing in Science and Engineering*, 14(4):18–25.

Heimann, M. and Keeling, C. D. (1989). *Climate Variability in the Pacific and the Western Americas*, chapter A three-dimensional model of atmospheric CO2 transport based on observed winds: 2. Model description and simulated tracer experiments, pages 237–275. American Geophysical Union.

McCallum, I., Wagner, W., Schmullius, C., Shvidenko, A., Obersteiner, M., Fritz, S., and Nilsson, S. (2009). Satellite-based terrestrial production efficiency modeling. *Carbon Balance and Management*, 4(1):8.

Perez, F., Granger, B. E., and Hunter, J. D. (2011). Python: An ecosystem for scientific computing. *Computing in Science and Engineering*, 13(2):13–21.

Ruimy, A., Dedieu, G., and Saugier, B. (1994). Methodology for the estimation of terrestrial net primary production from remotely sensed data. *J. Geophys. Res.*, 99:5263–5284.

Running, S., Nemani, R., Glassy, J., and Thornton, P. (1999). Modis daily photosynthesis (psn) and annual net primary production (npp) product (mod17). *Algorithm Theoretical Basis Document (ATBD), Version 30*.

Verstraeten, W. W., Veroustraete, F., and Feyen, J. (2006). On temperature and water limitation of net ecosystem productivity: Implementation in the c-fix model. *Ecological Modelling*, 199(1):4 – 22.

Voinov, A., Hood, R., Daues, J., Assaf, H., and Stewart, R. (2008). Chapter twelve building a community modelling and information sharing culture. In A.J. Jakeman, A.A. Voinov, A. R. and Chen, S., editors, *Environmental Modelling, Software and Decision Support*, volume 3 of *Developments in Integrated Environmental Assessment*, pages 345 – 366. Elsevier.