HUME: AN OBJECT ORIENTED COMPONENT LIBRARY FOR GENERIC MODULAR MODELLING OF DYNAMIC SYSTEMS

H. Kage & H. Stützel

Institute for Vegetable & Fruit Crops, Univ. Hannover, Herrenhaeuser Str. 2, D-30419 Hannover, Germany, E-mail:kage@gem.uni-hannover.de

Introduction

The task of modelling agricultural systems calls for a proper analysis of the system, for gathering of knowledge from different disciplines and for the collection of suitable data. But also of crucial importance are appropriate methods and tools of software technology for implementing these often quite complex models (Acock and Reynolds, 1997). Modularity and genericness are key words, which describe the now widely accepted strategy to overcome the problems of complexity which arise when constructing, maintaining and reusing models as a whole or in parts (Reynolds and Acock, 1997).

Agricultural like other dynamic systems may be analysed in terms of state variables, flows and auxiliary variables (De Wit, 1982). This usually ends up with a set of differential equations, which have to be integrated numerically, due to the complex structure of the systems and the irregular boundary conditions we normally have. The common structure makes it possible to support the whole process of designing and coding a model together with the technical tasks of input and output with appropriate software tools. This may be done using code libraries in 'classic' programming languages like FORTRAN, Pascal or C++. But there are also specially designed simulation programming languages available, like CSMP and ACSL (MGA-software, 1999) as well as a couple of integrated software programs, like Stella (HPS-Inc., 1999) and Modelmaker (Cherwell_Scientific_Inc., 1999). Also, crop growth models built in an object oriented design have been implemented from which classes may be used as templates for new models (Chen and Reynolds, 1997; Gauthier *et al*., 1999).

Measures for comparing these software tools are time effort for coding, portability, embedded structuring of code, speed of program execution and universality. Generally, there seems to be a trade-off between the level of support and the 'universality' one has within these support systems. All tools, however, underlie a continuously evolution, since they are strongly dependent on the soft- and hardware technology they rely on. The 'optimal' software tool differs therefore from person to person and changes in time due to further progress in hardware and software technologies.

Methods

Our approach, the HUME modelling environment is an object oriented class library based upon the Delphi^{TM} /C++ Builder^{TM} software development technology, which integrates an object pascal/C++ compiler and a comprehensive class library, the visual component library (VCL) for standard programming tasks into a visual programming environment. Modularity of software design is strongly supported by the concept of 'components', which are essentially well defined code entities which can be made easily accessible by positioning them into toolbars of the modelling environment. They are easily manipulated at 'design time' by means of an 'object inspector' without the necessity to go into the code of these components itself.

Results

Any model based on this library consists of one main model module, implemented in a class called ‘Tmod’ and a number of sub-models. The main model is responsible for the control of the simulation, single or multiple runs, and also implements methods like calculating basic statistics and parameter estimation based on the Levenberg-Marquardt method. All sub-models have to be derived from the base class ‘TsubMod’ which contains dynamic lists of state variables, variables, parameters and ‘external values’, i.e. values needed from outside the sub-model. The information exchange between the sub-models through ‘external values’ is flexible, since it is simply based on string identities between the information needed and information located in any other sub-model or input file. This technique allows exchange of sub-models through ‘drag and drop’ without any changes in the source code even for a changing number and order of parameters, as long as the necessary input parameters to the sub-model can be found anywhere else in the model. A graphical user interface based on the general data structure supports control of parameter values, initial values and allows input of measured data.

Based on these fundamental classes a component hierarchy has been and is still further developed, including several components for dry matter production, plant development, dry matter partitioning, root growth of plants as well as modules for soil water and soil nitrogen budget.

Conclusions

Efficient dynamic modelling of agricultural systems can strongly be supported by appropriate software tools. The choice of the appropriate tool, however depends on the kind of problem to be solved and on the personal background. The HUME component library is still under construction and not yet documented, but it may already be useful for ‘modellers’ as a source of components or as an inspiration for own projects.

References

Acock, B., and Reynolds, J. F. (1997). Introduction: modularity in plant models. Ecological modelling **94**, 1-6.

Chen, J. L., and Reynolds, J. F. (1997). GePSi: A generic plant simulator based on object-oriented principles. Ecological modelling **94**, 53-66.

Cherwell_Scientific_Inc. (1999). http://www.cherwell.com.

De Wit, C. T. (1982). Simulation of living systems. *In* “Simulation of plant growth and crop production. Ed.:.F.W.T. Penning de Vries, H.H. van Laar, pp. 3-7. Pudoc, Wageningen.

Gauthier, L., Gary, C., and Zekki, H. (1999). GPSF: a generic and object-oriented framework for crop simulation. Ecological modelling **116**, 253-268.

HPS-Inc. (1999). http://www.hps-inc.com.

MGA-software (1999). http://www.mga.com.

Reynolds, J. F., and Acock, B. (1997). Modularity and genericness in plant and ecosystem models. Ecological modelling **94**, 7-16.