ELSEVIER

Ecological Modelling 94 (1997) 53-66

ECOLOGICAL MODELLING

GePSi: A generic plant simulator based on object-oriented principles

Jia-Lin Chen*, James F. Reynolds

Department of Botany, Duke University, Durham, NC 27708-0338, USA

Abstract

The Generic Plant Simulator (GePSi) is a physiologically-based model that combines modules for canopy, root environment, water relations, and potential growth to generate whole-plant carbon, nitrogen, and water balances. The version presented here is coded in the object-oriented programming (OOP) language, C++, to enhance the implementation of modularity. In the aboveground aerial environment, the Weather module defines the weather conditions above a canopy, and MicroWeather defines the vertical profiles of micro-meteorological variables in a canopy. The belowground soil environment contains the SoilProperty modules, which define vertical profiles of physical and chemical variables in a soil column. The 'part-of' hierarchy in GePSi follows the structure of a real plant: the Plant module calls canopy and root system modules; the Canopy module, in turn, calls leaf, stem and fruit modules; and the RootSystem module calls coarse and fine root modules, etc. Our long-term goal is for GePSi to serve as a template for building a plant growth simulator by simply selecting appropriate modules for the question being asked. We are building a suite of plant modules (and their interfaces) based on general principles that are fundamentally similar for different kinds of plants. This includes photosynthesis, growth, nutrient and carbon allocation, water uptake, etc. These modules can be parameterized for specific species, related groups of species, life-forms, or broader groups depending on how variable the processes are across the groupings and the amount of unexplained variability that is acceptable for the question being investigated. Our modular-based approach has numerous advantages, including improving the understanding of the model, reducing duplication of effort, and facilitating the adaptation of the model for different sites and ecosystems.

Keywords: Modular; Plant growth; Simulation; Elevated CO_2; Hierarchy

1. Introduction

Plant growth and development has been modeled for more than three decades, resulting in a wide diversity of models and modeling approaches (see reviews by Kirschbaum et al., 1994; Leenhardt et al., 1995; Perrin and Sibly, 1993; Reynolds et al., 1996; Schuepp, 1993; Whisler et al., 1986; Yan and Wallace, 1996). While models have been developed for many types of species—

* Corresponding author.

0304-3800/97/$17.00
PII S0304-3800(96)01928-X

ranging from annual crops to long-lived trees and have included many unique characteristics, e.g. the ability to fix nitrogen, drought deciduousness, C4 photosynthetic pathway, etc—the overall similarity in the structure of these plant growth models is striking (Reynolds and Acock, 1985; Reynolds et al., 1986). This motivated us to propose the need for, and advantages of, a generic structure for plant growth models (Acock and Reynolds, 1989, 1990; Reynolds and Acock, 1985; Reynolds et al., 1989, 1993).

In this paper we report on our efforts to develop a generic, plant simulator (GePSi). GePSi is a physiologically-based model initially programmed in the Advanced Continuous Simulation Language (ACSL, 1987) to simulate the photosynthesis and growth of shrubs and trees (Reynolds and Cunningham, 1981; Reynolds et al., 1980). The version presented here is coded in the object-oriented programming (OOP) language, C++ (Symantec, 1995), in order to facilitate the implementation of modularity. A number of researchers have recently adopted the OOP approach for modeling ecological processes (e.g. Baveco and Lingman, 1992; Ferreira, 1995; Larkin et al., 1988; Laval, 1995; Sekine et al., 1991) and for plant growth modeling (e.g. Kolstrom, 1991; Sequeira et al., 1990, 1991, 1993; Väisänen et al., 1994; Van Evert and Campbell, 1994). Our primary motivation for using the OOPs paradigm is to establish a framework that will: (i) make it easier to maintain the model from year to year as various researchers contribute to its development; (ii) confine errors to individual modules in order to reduce programming efforts; (iii) allow easy replacement or exchange of process modules developed in our lab and elsewhere and (iv) facilitate the adaptation of the model to a variety of plants and ecosystem types. The last objective is related to our efforts to develop models to predict the effects of elevated CO_2 and climate change on plants and ecosystems (see Luo and Reynolds, 1997; Reynolds et al., 1992, 1996). A secondary objective is to contribute to current efforts to establish a generic, modular, structural design for plant growth models (see Reynolds and Acock, 1997).

The structure we present here is a result of much trial and error and will continue to evolve.

2. Class structure

The model consists of three sections: CONTROL, MODEL and SHOW (Fig. 1). CONTROL and SHOW consist of C++ class libraries (version 6.0, Symantec, 1995) that are machine specific whereas MODEL is portable to any operating system running C++. CONTROL allows the user to set the length of time to be simulated, to change parameter values, to set initial conditions, etc. SHOW displays results, e.g. by plotting graphs, writing output files, etc.

MODEL consists of three parts: ABIOTIC, BIOTIC, and TIMER. These parts and the classes they contain are illustrated in Fig. 1, which shows how the classes are related in a 'part-of' hierarchy

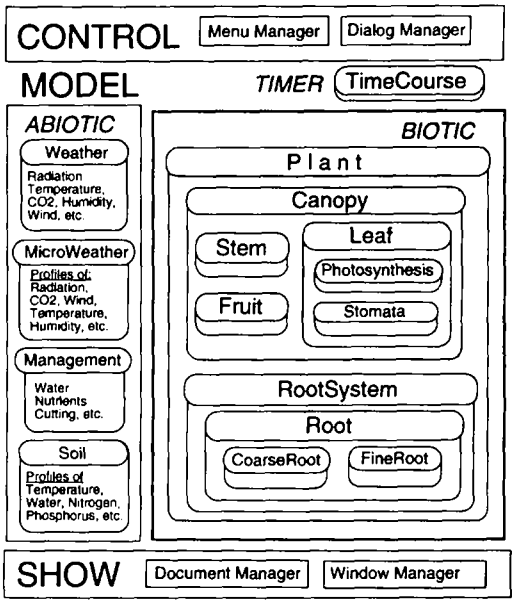

Fig. 1. GePSi is composed of three parts: CONTROL, MODEL, and SHOW. Menu Manager, Dialog Manager, Document Manager, and Window Manager are C++ class libraries that provide interfacing between MODEL and the user (in CONTROL) and display and output options (in SHOW). Classes are related in a part-of hierarchy, i.e. Canopy is a part-of Plant, etc. Not all classes shown.

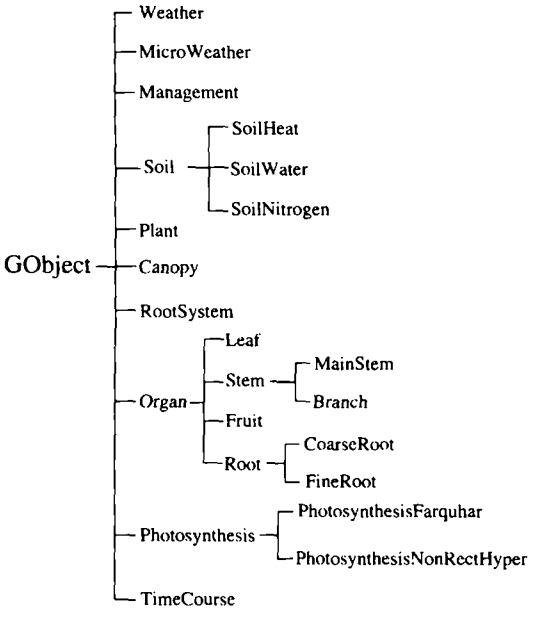

Fig. 2. Class hierarchy (i.e. the kind-of hierarchy) in GePSi. Illustrates inheritance relationships between classes. Not all classes shown.

(OOP terms defined in Acock and Reynolds, 1997). This also serves as the calling structure in GePSi, i.e. Plant calls Canopy, which calls Leaf, etc. ABIOTIC contains classes that determine driving variables for both the above- and belowground environment. The aboveground aerial environment contains two classes, Weather (conditions above a canopy) and MicroWeather (vertical profiles of micro-meteorological variables within a canopy). The belowground environment contains one class, SoilProperty — which has three subclasses: SoilHeat, SoilWater and SoilNitrogen to compute profiles of physical and chemical variables in a soil column. BIOTIC contains the classes that define plant growth. Whereas many models lump canopy processes (e.g. Norman, 1982) and micrometeorological processes (e.g. Goudriaan, 1977), we found that keeping them separate simplified the implementation of modularity in GePSi.

At execution time, the classes in GePSi are used to create (instantiate) objects to simulate plant growth. For example, we can create multiple plant objects from Plant if we are interested in simulating competition between individuals. Once a Plant object is created, it creates Canopy and RootSystem objects. The Canopy object, in turn, creates Leaf, Stem, and Fruit objects and the RootSystem object creates a Root object, which creates CoarseRoot and FineRoot objects, etc. Of course, the number of objects of Leaf, Stem, Fruit and Root may change during growth; when the Canopy and RootSystem objects update themselves, they may create new objects or destroy old ones. This is a relatively simple task in OOP.

Because of the calling structure, the use of the class Plant obligates the execution of all classes in the BIOTIC part of the model (Fig. 1). Weather can be used as a stand-alone model to analyze annual, daily and diurnal patterns of the weather generated from theoretical and empirical formulas. Other classes can be combined in various ways to address specific problems. For example, MicroWeather incorporates the full version of Norman (1982, 1993) model of radiation transfer and, when combined with Canopy, can be used by micrometeorologists to analyze profiles of radiation, temperature, wind speed, etc. in plant canopies without considering other plant processes. The class Canopy, of course, obligates the execution of classes Leaf, Stem and Fruit (Fig. 1).

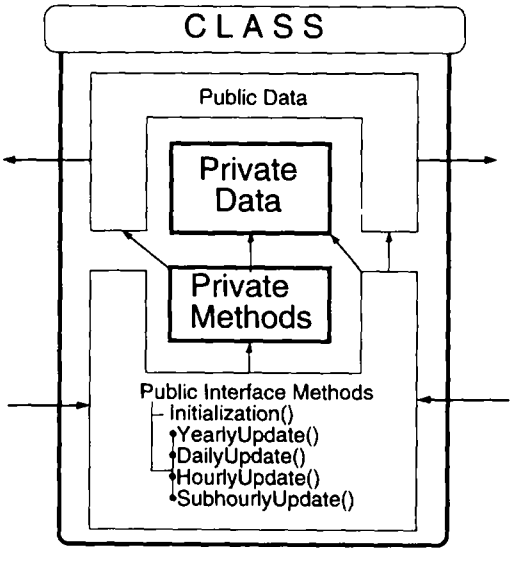

Fig. 3. Each class (object) has instance variables (i.e., data) and methods (i.e. operations) that are either public or private.

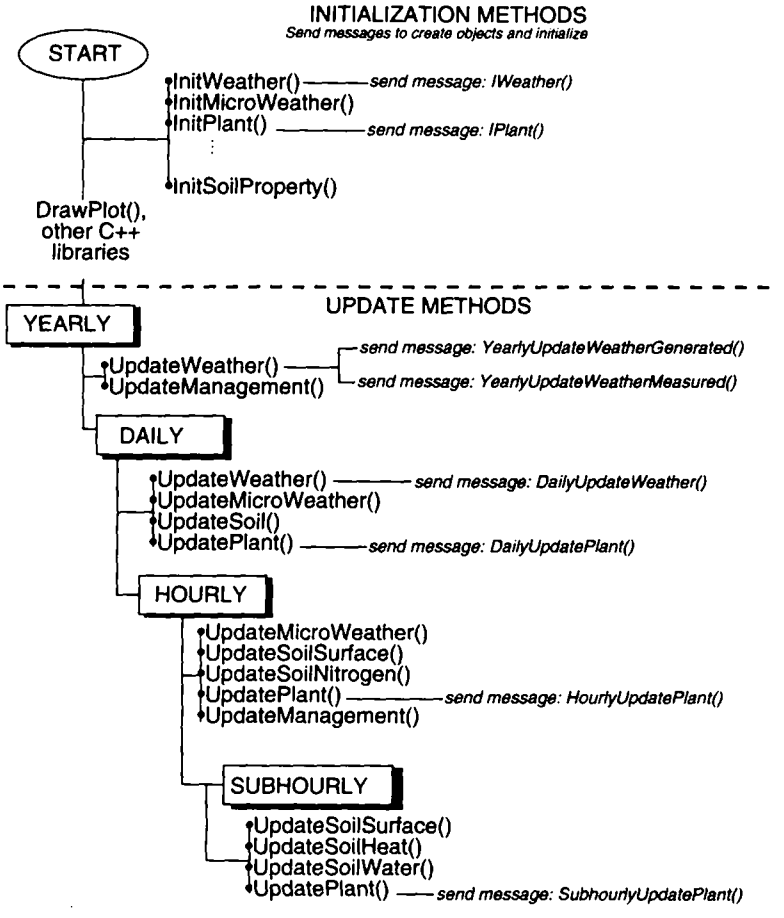

Fig. 4. Class TimeCourse, showing two types of public interface methods: *initialization* and *update*.

SoilHeat, SoilWater, and SoilNitrogen are based on partial differential equations for heat, water and nutrient movement in soils, respectively, and can be used to analyze soil data. Similarly, Leaf can be used by plant physiologists to examine leaf photosynthesis, stomatal conductance, and transpiration without referring to other classes in GePSi.

TIMER controls all of the time loops in MODEL. The class TimeCourse contains a method that explicitly defines a procedure to simulate the growth of a plant (described below). This introduction of procedure is at variance with the paradigm for object-oriented design (OOD) (see discussion in Acock and Reddy, 1997).

3. Class hierarchy

The class hierarchy (i.e. 'kind-of' hierarchy) for GePSi in Fig. 2 shows the inheritance relationships between classes (akin to a family tree). It is distinct from the calling structure (the 'part-of' hierarchy) shown in Fig. 1. In OOP, each class can inherit methods (i.e. operations) and attributes (i.e. data) from its superclass. Inheritance is one of the essential features of OOP: the ability to design a class with general features and derive specific cases from it. For instance, the class Organ (Fig. 2) contains shared characteristics of leaves, stems, fruits, and roots. By making the Leaf class a subclass of Organ, Leaf inherits those

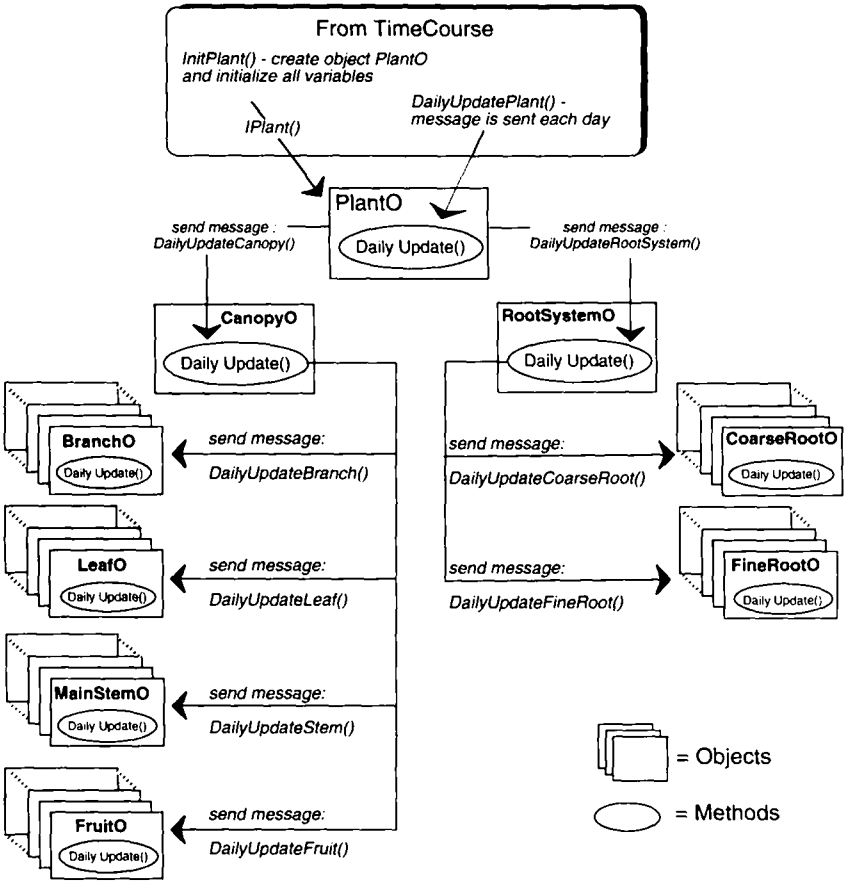

Fig. 5. Sequence of events in creation of objects, initialization, and updating for the plant object, PlantO.

shared characteristics. To complete the Leaf class, it is then only necessary to add those features that make leaves distinct from stems, fruits, and roots. This inheritance feature of OOP greatly facilitates the development of a truly generic model.

The class GObject (Fig. 2) is an abstract class derived from the root (i.e. the most basic) class of the C++ package (CObject in SYMANTEC C++, and TObject in BORLAND C++). All other classes in Fig. 2 are descendants of GObject and inherit all methods and attributes. This class provides the methods for the creation and destruction of objects, and is introduced to improve the portability of GePSi between different versions of the C++ language. The second hierarchical level includes classes such as Weather, Plant, Canopy, Organ, RootSystem, and Soil (Fig. 2). Leaf, Stem, Fruit, and Root are subclasses of Organ, at the third hierarchical level.

4. Interface methods

Each class in MODEL has attributes and methods that are either public or private. Public connotes that they are accessible from outside of the class, while private means they are not accessible (Fig. 3).

We established two types of public interface methods: *initialization* and *update*. These methods are invoked by sending a *message* to the appropriate object, in effect instructing the object to execute the method. Initialization methods create objects from classes. All messages in GePSi are

Table 1
Interface methods associated with a selection of classes that generate objects used in GePSi. These are shown to illustrate some of the input-output variables considered in the model

Class	Interface methods	Input variables required	Variables calculated or updated
Weather	IWeather()	Latitude Longitude Standard meridian Reference height Atmospheric pressure A flag specifying if daily or hourly measured data available	
	YearlyUpdate-WeatherGen-erated()	Annual mean/amplitude of air temperature	Annual mean and amplitude of air temperature
		Amplitude of daily air temperature Average air relative humidity on sunny days Annual mean wind speed Annual mean atmospheric CO_2 concentration Annual total precipitation Sunny or rainy day (each of 365 days)	Amplitude of daily air temperature change Average air relative humidity on sunny days Annual mean wind speed Annual mean atmospheric CO_2 concentration Daily precipitation on rainy days Sunny or rainy day (per year)
	YearlyUpdate-WeatherMea-sured()	File name of input data	
	DailyUpdate-Weather()	Julian day File name of the input data	Short-wave radiation (per hour) Direct and diffuse PAR and NIR (per hour) Air temperature (per hour) Relative humidity (per hour) Wind speed (per hour) Soil surface temperature (per hour) Precipitation (per hour)
MicroWeather	IMicroWeather()	Number of inclination classes in leaf angle distribution function Leaf angle distribution function Cluster factor Leaf reflectance to PAR and NIR Leaf transmittance to PAR and NIR Soil reflectance and transmittance to PAR and NIR	Leaf projections to each sky zone
	DailyUpdate-MicroWeather()	Number of canopy layers Height of each layer Leaf and stem area in each layer	Height of each layer Leaf and stem area in each layer Extinction factors to diffuse radiation
	HourlyUpdate-MicroWeather()	Air pressure above canopy Sun zenith angle CO_2 concentration above the canopy	Profiles of CO_2 concentration Profile of wind speed Profiles of air temperature and vapor pressure
		Direct PAR and NIR above the canopy Downward diffuse PAR and NIR above canopy	Profiles of direct PAR and NIR Profiles of downward diffuse PAR and NIR
		Thermal radiation from sky Air temperature above canopy	Profiles of upward diffuse PAR and NIR Profiles of downward and upward thermal radiation

Table 1 (continued)

Class	Interface methods	Input variables required	Variables calculated or updated
		Air vapor pressure above canopy Wind speed above canopy Soil surface temperature	NO_3 concentration in irrigation water
Soil	ISoil()	Number of soil types in column Depth of each soil type Soil type index Number of layers Maximum depth Index for layer thickness increase Initial profile of soil temperature Initial profile of soil water potential or water content Initial profiles of concentrations of NH_4 and NO_3	
	DailyUpdate-Soil()	Number of soil layer with roots	Number of soil layer with roots
	HourlyUpdate-Heat()	Air pressure Radiation at soil surface (PAR, NIR, thermal) Air temperature Air vapor pressure Wind speed Soil temperature at bottom of the column Time step of integration	Profiles of soil temperature Sensible heat fluxes at the surface Latent heat fluxes at the surface
	HourlyUpdate-Nutrients()	Concentration of NH_4 and NO_3 in the irrigation water Root uptake rate of NH_4 and NO_3 in each layer	Profiles of NH_4 and NO_3
	SubhourUpdate-Water()	Precipitation/irrigation rate at soil surface Water potential or content at bottom of soil column Root uptake of water in each layer Time step of integration	Profile of water potential Profile of volumetric water content Water fluxes between layers
Plant	IPlant()	Data file name of canopy File name of root-system Number of soil layers Bottom depth of each soil layer Flag for water dynamics Flag for nutrient dynamics	Canopy properties (see CANOPY) Root-system properties (see ROOTSYSTEM)
	DailyUpdate-Plant()	Daily mean air temperature Daily mean soil temperature	Plant age Carbon allocation to canopy and root Nitrogen allocation to canopy and root Canopy properties (see CANOPY) Root-system properties (see ROOTSYSTEM)
	HourlyUpdate-Plant()	Soil temperature in each layer Concentration of NH_4 and NO_3 in each layer	Canopy properties (see CANOPY) Root-system properties (see CANOPY)

Table 1 (continued)

Class	Interface methods	Input variables required	Variables calculated or updated
		Dispersion coefficients of NH_4 and NO_3 in soil	
	Subhourly-UpdatePlant()	Hydraulic conductivity in each soil layer Water potential in each soil layer	Canopy properties (see CANOPY) Root-system properties (see ROOTSYSTEM)
		Time step for integration	
Canopy	ICanopy()	File name of input data Number of layers and flushes Covered ground area Leaf inclination distribution Air pressure Growth CO_2 concentration Tolerance for leaf energy balance Average leaf nitrogen content Water potential of the canopy Age, top diameter, biomass of main stems in each layer Age, top diameter, biomass of branches in each layer and flush Age, length, width, biomass of leaves in each layer and flush Flag for water dynamics Flag for nutrient dynamics Pointer to MicroWeather object	
	DailyUpdate-Canopy()	Daily mean air temperature	Carbon and nitrogen supply to each stem, leaf and fruit
		Daily carbon supply	Heat sum for emergence of new stem, leaf and fruit
		Daily nitrogen supply	Total biomass in stems, leaves and fruits Total nitrogen in stems, leaves and fruits Total leaf area and stem area Daily respiration rates of stems, leaves and fruits Daily photosynthesis and transpiration of all leaves Canopy height and area
	HourlyUpdate-Canopy()		Hourly respiration rates of stems, leaves and fruits Photosynthesis and transpiration of sunlit leaves Photosynthesis and transpiration of shaded leaves Hourly gross and net assimilation of the canopy Hourly total transpiration
	SubhourUpdate-Canopy()	Time step for integration	Water potential and relative water content of the canopy
		Water uptake by roots	Total amount of water in the canopy

Table 1 (continued)

Class	Interface methods	Input variables required	Variables calculated or updated
RootSystem	IRootSystem()	Numbers of layers with roots	Age, biomass, length, diameter, water potential, nitrogen content of the suberized root
		Bottom depth of each layer	Age, biomass, length, diameter, water potential, nitrogen content of active roots in each layer and class
		Flag for water dynamics	
		Flag for nutrient dynamics	
		File name of the input data	
		Maximum ground area	
		Covered ground area	
		Number of root classes	
		Maximum rooting depth	
		Initial rooting depth	
		Age	
		Total biomass of roots	
		Average diameter of the suberized root	
	DailyUpdate-RootSystem()	Daily mean soil temperature	Carbon and nitrogen supply to each root
		Carbon supply to the root system	Rooting depth
		Nitrogen supply to the root system	Total biomass in suberized and active roots
			Total nitrogen in suberized and active roots
			Total root length
			Daily respiration rates of suberized and active roots
			Daily total water and nutrient uptake by roots
	HourlyUpdate-RootSystem()	Soil temperature in each layer	Hourly respiration rates of suberized and active roots
		Concentration of NH_4 and NO_3 in each layer	Hourly average water uptake by roots
		Dispersion coefficients of NH_4 and NO_3 in soil	Hourly nutrient uptake by roots
	SubhourUpdate-RootSystem()	Hydraulic conductivity in each soil layer	Water uptake by roots in each layer
		Water potential in each soil layer	
		Water potential of the canopy	
		Time step for integration	
Leaf	ILeaf()	Age	
		Heat sum for mature	
		Heat sum for death	
		Dry weight	
		Non-structural carbon	
		Length	
		Width	
		Water potential	
		Temperature	
		Nitrogen content	
		Growth CO_2 concentration	
		Air pressure	
		Tolerance for leaf energy balance	
		Parameters for Ball's model	
		Parameters for Farquhar's model	

Table 1 (continued)

Class	Interface methods	Input variables required	Variables calculated or updated
	DailyUpdate-Leaf()	Daily mean air temperature Carbon supply Nitrogen supply	Age and physiological age Dry weight Nitrogen content Daily total respiration Non-structural carbon Daily total assimilation Surface area
	HourlyUpdate-Leaf()	Air CO_2 PAR, NIR and thermal radiation Air temperature Vapor pressure Wind speed	Leaf-boundary-layer conductance Stomatal conductance Photosynthetic rate Transpiration rate Day respiration rate Maintenance respiration rate Intercellular CO_2 concentration
Branch	IBranch()	Age Heat sum for mature Heat sum for death Dry weight Non-structural carbon Length Width Water potential Temperature Nitrogen content Growth CO_2 concentration Tip angle Taper	
	DailyUpdate-Branch()	Daily mean air temperature Carbon supply Nitrogen supply	Age and physiological age Dry weight Nitrogen content Daily total respiration Non-structural carbon Active biomass Volume Top and base diameters
	HourlyUpdate-Branch()	Air temperature	Maintenance respiration
Fruit	IFruit()	Age Heat sum for mature Heat sum for death Dry weight Non-structural carbon Length Width Water potential Temperature Nitrogen content	
	DailyUpdate-Fruit()	Daily mean air temperature Carbon supply Nitrogen supply	Age and physiological age Dry weight Nitrogen content

Table 1 (continued)

Class	Interface methods	Input variables required	Variables calculated or updated
FineRoot	IFineRoot()	Age Heat sum for mature Heat sum for death Dry weight Non-structural carbon Length Width Water potential Temperature Nitrogen content Surface conductance to water Parameters of Michaelis-Menton function for NH_4 and NO_3 uptake	
	DailyUpdate-FineRoot()	Daily mean soil temperature Carbon supply Nitrogen supply	Age and physiological age Dry weight Nitrogen content Daily total respiration Non-structural carbon Daily suberization rate Daily water uptake per root length
	HourlyUpdate-FineRoot()	Soil temperature Soil NH_4 and NO_3 concentration Soil resistance to NH_4 and NO_3 transfer	Maintenance respiration Root-surface resistance to water NH_4 uptake rate NO_3 uptake rate
	SubhourlyUpdate FineRoot()	Time step Soil resistance to water Soil water potential Canopy water potential	Water uptake rate
CoarseRoot	ICoarseRoot()	Age Heat sum for mature Heat sum for death Dry weight Non-structural carbon Length Width Water potential Temperature Nitrogen content Surface conductance to water	
	DailyUpdate-CoarseRoot()	Daily mean soil temperature Carbon supply Nitrogen supply	Age and physiological age Dry weight Nitrogen content Daily total respiration Non-structural carbon Active biomass Daily suberization rate Daily water uptake per root length
	HourlyUpdate-CoarseRoot()	Soil temperature	Maintenance respiration

identified by having the extension '()' e.g. IPlant(), whereas all objects have the extension 'O', e.g. PlantO. For example, sending the message IWeather() to class Weather will create a object, WeatherO, and sending the message IPlant() to class Plant, will create an object, PlantO (Fig. 4). The creation of PlantO automatically results in the creation of CanopyO and RootsystemO, which in turn, causes the creation of objects for branches, leaves, roots, etc. (Fig. 5).

We have incorporated four update methods into the classes (Fig. 4). These correspond to time scales of yearly, daily, hourly and subhourly. Each object can be updated with a different frequency. For example, WeatherO is updated at the start of each year by method YearlyUpdateWeatherGenerated(), if the weather is being generated by mathematical relations, or by YearlyUpdateWeatherMeasured() if being read in from a data file. PlantO is updated subhourly, hourly, and daily depending on the application. The sequence of events enacted during a daily update of object PlantO are illustrated in Fig. 5. During the growth of a plant, the update methods increases the size, weight and/or nitrogen content of each organ, and creates new objects, depending on environmental conditions.

Although Fig. 4 appears similar to a procedural flow chart, there are some important differences. The messages in Fig. 4 are only instructions to an object to invoke a method. In a Fortran program, for example, these would be calls to subroutines.

In Table 1, we provide some details of the initialization and update methods, and the variables involved in each class of object in GePSi. This table provides the reader with an overview of some of the processes we consider in the model.

5. Summary

Our goal in developing the GePSi is to build a modular structure based on general principles, and to write a suite of modules with defined interfaces, that are fundamentally similar for different kinds of plants. These modules include photosynthesis, growth, nutrient and carbon allocation, water uptake, etc. GePSi can be used to study a variety of important plant-level issues that have implications for understanding the carbon balance of ecosystems. This includes, for example, non-structural carbohydrate accumulation in plants that decouples photosynthesis from growth; changes in litterfall; root exudation and increased turnover rates; altered respiration rates; and allocation of biomass into above- and below-ground structures. These modules can be parameterized for specific species, related groups of species, life-forms, or broader groups depending on how variable the processes are across the groupings and the amount of unexplained variability that is acceptable for the question being investigated. This approach also allows the development, testing, and validation of each of the individual modules. Our ultimate goal is that GePSi can serve as a template for building a plant growth model by simply selecting the appropriate modules for the question being asked and either re-parameterizing or modifying them as described in Luo et al. (1997).

Acknowledgements

This research was supported by DOE grant DE-FG05-92ER61493, NSF grant DEB-9524058 and it is a contribution to the Jornada LTER under NSF grant DEB 92-40261. Additional support provided by the Forest-Atmosphere Carbon Transfer and Storage (FACTS-1) project at Duke University under DOE contract numbers DE-AC02-76H00016 at Brookhaven National Laboratory and DE-FG05-95ER62083 at Duke University.

References

Acock, B. and Reddy, V.R., 1997. Designing an object-oriented structure for crop models. Ecol Model., 94: 33–44.

Acock, B. and Reynolds, J.F., 1989. The rationale for adopting a modular generic structure for crop simulators. Acta Hortic., 248: 391–396.

Acock, B. and Reynolds, J.F., 1990. Model structure and database development. In: R.K. Dixon, R.S. Meldahl, G.A. Ruark and W.G. Warren (Editors), Process Model-

ing of Forest Growth Responses to Environmental Stress. Timber Press, Portland, Oregon, pp. 169–179.

Acock, B. and Reynolds, J.F., 1997. Introduction: Modularity in plant growth models. Ecol. Model., 94: 1–6.

ACSL, 1987. Advanced Continuous Simulation Language, Reference Manual. Report No. Vers. 4.2, Mitchell and Gauthier, Concord, MA.

Baveco, J.M. and Lingman, R., 1992. An object-oriented tool for individual-oriented simulation: host-parasitoid system application. Ecol. Model., 61: 267–286.

Ferreira, J.G., 1995. ECOWIN—An object-oriented ecological model for aquatic ecosystems. Ecol Model, 79: 21 34.

Goudriaan, J., 1977. Crop Micrometeorology: A Simulation Study. Pudoc, Wageningen. 249 pp.

Kirschbaum, M.U.F., King, D.A., Comins, H.N., McMurtrie, R.E., Medlyn, B.E., Pongracic, S., Murty, D., Keith, H., Raison, R.J., Khanna, P.K. and Sheriff, D.W., 1994. Modelling forest response to increasing CO2 concentration under nutrient-limited conditions. Plant Cell Environ, 17: 1081 1099.

Kolstrom, T., 1991. Modelling early development of a planted pine stand: an application of object-oriented programming. For. Ecol. Manage, 42: 63–78.

Larkin, T.S., Carruthers, R.I. and Soper, R.S., 1988. Simulation and object-oriented programming: the development of SERB. Simulation, 52: 93–100.

Laval, P., 1995. Hierarchical object-oriented design of a concurrent, individual-based, model of a pelagic tunicate bloom. Ecol Model, 82: 265–276.

Leenhardt, D., Voltz, M. and Rambal, S., 1995. A survey of several agroclimatic soil water balance models with reference to their spatial application. Eur J Agron, 4: 1 14.

Luo, Y., Field, C.B. and Mooney, H.A., 1997. Adapting GePSi (Generic Plant Simulator) for modeling studies in the Jasper Ridge CO_2 project. Ecol. Model., 94: 81–88.

Luo, Y. and Reynolds, J.F., 1997. A conceptual framework for studying ecosystem carbon cycling in response to rising CO2 concentration. submitted.

Norman, J.M., 1982. Simulation of microclimates. In: J. Hatfield and I.J. Thomason (Editors), Biometerology in Integrated Pest Management. Academic Press, NY, pp. 65–99.

Norman, J.M., 1993. Scaling processes between leaf and canopy levels. In: J. Ehleringer and C. Field (Editors), Scaling Processes Between Leaf and the Globe. Academic Press, NY, pp. 41–76.

Perrin, N. and Sibly, R.M., 1993. Dynamic models of energy allocation and investment. Annu Rev Ecol Syst, 24: 379–410.

Reynolds, J.F. and Acock, B., 1985. Predicting the response of plants to increasing carbon dioxide: A critique of plant growth models. Ecol. Model., 29: 107–129.

Reynolds, J.F. and Acock, B., 1997. Modularity and genericness in plant and ecosystem models. Ecol. Model., 94: 7–16.

Reynolds, J.F., Acock, B., Dougherty, R. and Tenhunen, J.D., 1989. A modular structure for plant growth simulation models. In: J.S. Pereira and J.J. Landsberg (Editors), Biomass Production by Fast-Growing Trees, NATO ASI Series, Applied Science. Kluwer, Dordrecht, pp. 123–134.

Reynolds, J.F., Acock, B. and Whitney, R., 1993. Linking CO2 experiments and modeling. In: E.-D. Schulze and H.A. Mooney (Editors), Design and Execution of Experiments on CO2 Enrichment, Report No.6. Ecosystems Research Series of Environmental Research Programme, Commission of the European Communities, Brussels, pp. 93–106.

Reynolds, J.F., Bachelet, D., Leadley, P. and Moorhead, D., 1986. Assessing the effects of elevated carbon dioxide on plants: toward the development of a generic plant growth model. Report No. Progress Report 023, US Department of Energy, Washington, DC.

Reynolds, J.F. and Cunningham, G.L., 1981. Validation of a primary production model of the desert shrub *Larrea tridentata* using soil-moisture augmentation experiments. Oecologia, 51: 357–363.

Reynolds, J.F., Hilbert, D.W., Chen, J.-L., Harley, P.C., Kemp, P.R. and Leadley, P.W., 1992. Modeling the Response of Plants and Ecosystems to Elevated CO2 and Climate Change. DOE/ER-60490T-H1, National Technical Information Service, U.S. Dept. of Commerce, Springfield, VA. 220 pp.

Reynolds, J.F., Kemp, P.R., Acock, B., Chen, J.-L. and Moorhead, D.L., 1996. Limitations, uncertainties and challenges in modeling the effects of elevated CO2 on plants and ecosystems. In: G. Koch and H.A. Mooney (Editors), Carbon Dioxide and Terrestrial Ecosystems. Academic Press, San Diego, pp. 347–380.

Reynolds, J.F., Strain, B.R., Cunningham, G.L. and Knoerr, K.R., 1980. Predicting primary productivity for forest and desert ecosystem models. In: J.D. Hesketh and J.W. Jones (Editors), Predicting Photosynthesis for Ecosystem Models, Vol. II. CRC Press, Boca Raton, FL, pp. 169–207.

Schuepp, P.H., 1993. Tansley Review No. 59 – Leaf Boundary Layers. New Phytol, 125: 477–507.

Sekine, M., Nakanishi, H., Ukita, M. and Murakami, S., 1991. A shallow-sea ecological model using an object-oriented programming language. Ecol. Model., 57: 221–236.

Sequeira, R.A., Makela, M.E., El-Zik, K.M. and Stone, N.D., 1990. Coupling object-oriented plant models and Heliothis models. Beltwide Cotton Conference. Natl. Cotton Council of America: Las Vegas, NV, pp. 339-342.

Sequeira, R.A., Sharpe, P.J.H., Stone, N.D., El-Zik, K.M. and Makela, M.E., 1991. Object-oriented simulation: Plant growth and discrete organ to organ interactions. Ecol. Model., 58: 55–90.

Sequeira, R.A., Stone, N.D., Makela, M.E., El-Zik, K.M. and Sharpe, P.J.H., 1993. Generation of mechanistic variability in a process-based, object-oriented plant model. Ecol. Model., 67: 285 306.

Symantec, C., 1995. Symantec, C++, Vers. 6.0: Cupertino, CA.

Väisänen, H., Standman, H. and Kellomäki, S., 1994. A model

for simulating the effects of changing climate on the functioning and structure of the boreal forest ecosystem—an approach based on object-oriented design. Tree Physiol, 14: 1081–1095.

Van Evert, F.K. and Campbell, G.S., 1994. CropSyst—a collection of object-oriented simulation models of agricultural systems. Agron J, 86: 325–331.

Whisler, F.D., Acock, B., Baker, D.N., Fye, R.E., Hodges, H.F., Lambert, J.R., Lemmon, H.E., McKinion, J.M. and Reddy, V.R., 1986. Crop simulation models in agronomic systems. Adv. Agron., 40: 141–208.

Yan, W. and Wallace, D.H., 1996. A model of photoperiod x temperature interaction effects on plant development. Crit Rev Plant Sci, 15: 63–96.