Chapter 4
A Component-Based Framework for Simulating Agricultural Production and Externalities

Marcello Donatelli, Graham Russell, Andrea Emilio Rizzoli, Marco Acutis, Myriam Adam, Ioannis N. Athanasiadis, Matteo Balderacchi, Luca Bechini, Hatem Belhouchette, Gianni Bellocchi, Jacques-Eric Bergez, Marco Botta, Erik Braudeau, Simone Bregaglio, Laura Carlini, Eric Casellas, Florian Celette, Enrico Ceotto, Marie Hélène Charron-Moirez, Roberto Confalonieri, Marc Corbeels, Luca Criscuolo, Pablo Cruz, Andrea di Guardo, Domenico Ditto, Christian Dupraz, Michel Duru, Diego Fiorani, Antonella Gentile, Frank Ewert, Christian Gary, Ephrem Habyarimana, Claire Jouany, Kamel Kansou, Rob Knapen, Giovanni Lanza Filippi, Peter A. Leffelaar, Luisa Manici, Guillaume Martin, Pierre Martin, Eelco Meuter, Nora Mugueta, Rachmat Mulia, Meine van Noordwijk, Roelof Oomen, Alexandra Rosenmund, Vittorio Rossi, Francesca Salinari, Ariel Serrano, Andrea Sorce, Grégoire Vincent, Jean-Pierre Theau, Olivier Thérond, Marco Trevisan, Patrizia Trevisiol, Frits K. van Evert, Daniel Wallach, Jacques Wery, and Arezki Zerourou

Introduction

Several simulation tools allow the impact of agricultural management on production activities in specific environments to be studied (e.g. Brisson et al. 2003; Keating et al. 2003; Jones et al. 2003; Stockle et al. 2003; Van Ittersum et al. 2003). Such tools are specialized, to different extents, to one or more specific production activities: arable crops or cropping systems, grassland, orchards, agro-forestry, livestock etc. Their outputs often only include estimates of a restricted range of system externalities which may have a negative environmental impact; these may include, for example, nitrogen leaching or the fate of pesticides. Very often, the structure of such systems neither allows for an easy plug-in of models for new agricultural production activities, nor the use of different approaches for the simulation of processes via alternate formulations. Furthermore, documentation of such tools may not be up-to-date, and may not follow a single standard, which makes it difficult to access information.

M. Donatelli (✉)
CRA-CIN-Agriculture Research Council, Via di Corticella 133, 40128 Bologna, Italy
e-mail: marcello.donatelli@entecra.it.

F.M. Brouwer and M. van Ittersum (eds.), *Environmental and Agricultural Modelling: Integrated Approaches for Policy Impact Assessment,* DOI 10.1007/978-90-481-3619-3_4,

Finally, when such systems are proprietary systems of either research groups or projects, it may not be possible for third parties to re-use the system for further development.

A basic requirement of any biophysical model is that it must be able to simulate the processes which influence significantly the behaviour of a system, and particularly those aspects that relate to the purpose of the model. An obvious example is that the model must not be restricted to potential production if its intended use is to study water-limited agriculture This example, however, is at a "high" level, meaning that simulating water-limited production may require:

- The simulation of a different number of approaches; and
- Even different approaches for the simulation of the same process, when environmental conditions change

As an example of the first point, studying the impact of mulching requires soil evaporation models which react to soil cover beyond that given by canopy cover, and the fate of the mulching material must also be simulated. An example of the second point can be simulating the water budget of conditions typified by peak evapotranspiration of 5 mm day^{-1} on a deep soil compared with conditions where the peak evapotranspiration is 12 mm day^{-1} on a shallow, cracking soil. The former case can be simulated with simpler, yet still adequate, approaches compared to the latter. Moreover, some approaches may demand inputs which may not be easily available, thus compromising its operational use. Also, as peer reviewed publications may propose alternative options for modelling processes with the same assumptions; tests need to be carried out to assess performance and reliability throughout the range of operational conditions. Finally, effective simulation of a biophysical system, no matter what level of simplification is chosen to simulate its behaviour, requires expertise in different domains. This is a demanding task that requires a multi-team effort for system analysis and model development. All these reasons, argue for a flexible and modular simulation system, and provide, in effect, a specification for the simulation system described in this chapter.

The advent of component-based software engineering has enabled the development of scalable, robust, large-scale applications in a variety of domains, including agro-ecological modelling. In systems analysis, it is common to deal with the complexity of an entire system by considering it to consist of linked sub-systems. This leads naturally to thinking of models as being made of sub-models. Such a conceptual model can be implemented as a computer model composed of connected component models. This type of implementation has at least two major advantages. First, new models can be constructed by connecting existing component models of known and guaranteed quality to new component models. This has the potential to increase the speed of development. Secondly, the predictive capabilities of two different component models can be compared, as opposed to only comparing whole simulation systems. Further, common and frequently used functionalities, such as numerical integration, visualization and statistical ex-post analysis, can be implemented as generic tools which are developed once and shared by all the model developers.

In the past decade there has been an increasing demand for modularity and replaceability in biophysical models (e.g. Jones et al. 2001; David et al. 2002; Donatelli et al., 2003, 2004, 2006a), aimed both at improving the efficiency of use of resources and at fostering a higher quality of modelling through specialization of model builders in their specific domain. The modular approach developed in the software industry is based on the concept of encapsulating the solution of a modelling problem in discrete, replaceable, and interchangeable software units called components. A software component can be defined as "a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject by composition by third parties" (Szypersky et al. 2002). Component-oriented designs actually represent a natural choice for building scalable, robust, large-scale applications, and to maximize the ease of maintenance in a variety of domains, including agro-ecological modelling (Argent 2004). This concept has been applied to biophysical simulation and has led to the development of modelling frameworks such as Simile, MODCOM, IMA, TIME, OpenMI, SME, and OMS (Argent and Rizzoli 2004; Rizzoli et al. 2004), which allow use to be made of components by linking them either directly or through a simulation engine. Three major aspects of model implementation are specific to the modelling platform, demand consistent development resources, and are real barriers to reusability. These are:

- Data input/output procedures (e.g. input/output data handling and file management)
- Common services (e.g. state variable integrator, simulation event handler); and
- Graphical user interfaces (GUI)

Modelling frameworks can play a key role in addressing these issues. First, the framework allows the application-specific parts of simulations to be segregated from the code employed to accomplish common tasks, thus greatly enhancing code re-use (Hillyer et al. 2003). Secondly, by defining those elements of the framework that actually contain the model implementation and how the elements are used, a designer can be presented with a clear path from conceptual model to simulation (Hillyer et al. 2003). Furthermore, by avoiding the need to re-implement common services, resources can be concentrated on the development of simulation components.

Developing a simulation system based on the component-oriented paradigm poses specific challenges in terms of both biophysical model linkages and implementation architecture. Component-based architectures demand the definition and implementation of sub-systems which minimize the need for links to other components, and the need for repeated communication between components. However, even when a system to be simulated is divided into sub-systems with little need for communication between them, data exchange prior to integration within a time step is needed, thus an articulated interface is needed that allows such calls. Although being potentially prone to mix and match "everything" is often suggested as an intrinsic weakness of component-based systems, this problem can be overcome by shifting the focus to the components themselves using semantically rich interfaces which ensure that the linked variables are appropriate. To illustrate the concept, if a component makes available a variable characterized by units, range of use, type and description, and

another component requires the same variable as an input, the link can be considered correct if a check of the variable attributes show that these are identical, whereas the correctness of the variable as an input must be investigated within the component producing the output. The principle of applying "parsimony" is of course still valid in model building. For instance, there is no point in coupling two components in which strong assumptions (and thus the limitations) of one impose an unnecessary burden on the modelling capabilities of the other. This, however, applies both to monolithic and component-based system development. As always, the choice of model should be conditioned by both the intended application of the model and a comprehensive system analysis, and this is totally independent of the type of implementation.

The SEAMLESS project has developed a framework to integrate analyses of impacts on a wide range of aspects of sustainability and multi-functionality (Van Ittersum et al. 2008). This requires the evaluation of the agricultural outputs and system externalities for a wide range of production systems and environments. Although some indicators of system performance can be provided using static models derived from existing databases, estimating system behaviour for novel techniques or existing techniques applied to new environments requires process based simulation. Also, even for known systems, some of the externalities due to agricultural production are only available as observational data for a very small number of experimental sites. The analysis of the biophysical components of agricultural systems thus requires a simulation framework which can be extended and updated by research teams, which allows easy incorporation of research results into operational tools, and which is transparent with respect to its contents and its functionality. The problems and requirements outlined in the previous paragraph have formed the basis of the design of the Agricultural Production and Externalities Simulator (APES) which offers flexibility in being an open modelling environment that allows an extensible set of modelling choices. The emphasis in APES has been to provide a transparent and flexible modelling platform that can be adapted to different modelling goals. This is a quite different rationale from the specific biophysical modelling solutions that are currently implemented.

APES: The Agricultural Production and Externalities Simulator

APES is a simulation model system for estimating the biophysical behaviour of agricultural production systems in response to the interaction of weather, soil and agro-technical management options. The system allows the incorporation, at a later date, of other modules which might be needed to simulate processes not included in the existing version, such as the impact of plant pests (see also Fig. 4.1).

Biophysical processes are simulated in APES using deterministic approaches which are mostly based on mechanistic representations. The criteria to select modelling approaches were the need to:

- Account for specific processes to simulate soil-land use interactions
- Input data to run simulations, which may be a constraint at the European scale

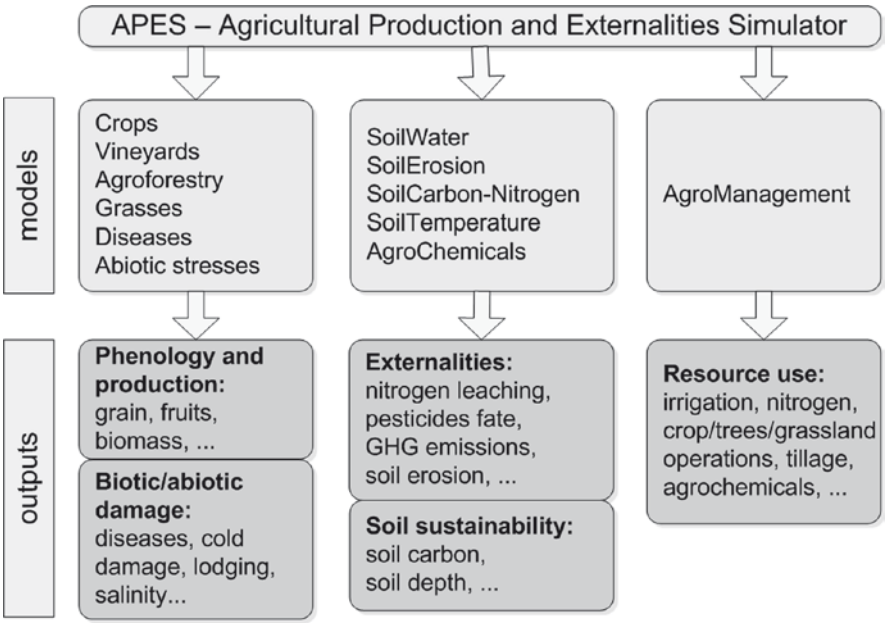

Fig. 4.1 Main typologies of models and outputs of APES. The details of both are described in the text

- Simulate all agricultural production activities of interest (e.g. crops, grasses, vineyards, agro-forestry); and
- Simulate agro-management and its impact on the system

Component Based Structure

There is no single solution to the problem of splitting complex systems into components. However, some divisions are more effective than others. The criteria used for doing this in APES were:

- Consistency with knowledge about the organization of the real system
- Consistency with the goal of encapsulating a useful/reusable set of modelling solutions relevant to the specific domain; and
- Minimization of the need for communication between components within a time step

This has led to components being developed with different model granularities (from the whole system perspective) as one of the possible solutions to modularization of agricultural production systems. Figure 4.2 shows the APES model components included in the December 2008 release.

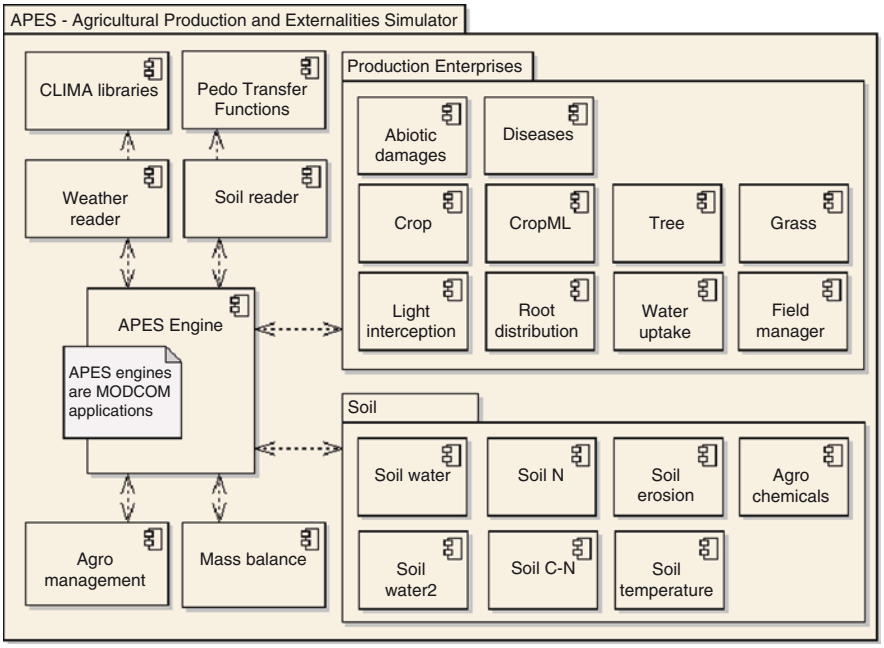

Fig. 4.2 The APES "coarse" component diagram. Note that there are alternate options for simulating soil water, soil nitrogen, and crops; also, within each of the components there can be alternate approaches for simulating processes

Model Components

APES is composed of two main groups of software units: the simulation engine which uses the modelling framework MODCOM (Hillyer et al. 2003), and the model components, which include a cross-component unit to compute mass balance. Model components can be grouped into agricultural management, soil components, production enterprise components, and weather.

The description of the models, implemented is available in the help files of each component (see "Web resources"). Help files are in general divided into two sections: "models", which contain the model description, targeted at model users, and "design and use" which contain component information targeted at developers. The components on public release also include a code documentation file and sample applications in a software development kit.

All models use a daily time step for integration and communication across modules, although calculations can be carried out with a shorter time step within a component. Each component contains one or more existing models which simulate the constituent processes. The relevant references are listed in the documentation of each component. A brief summary description of each component follows; the teams which have developed each component are detailed (refer to authors' affiliation for explanation of the acronyms). Components are grouped with reference to Fig. 4.2.

Agro-management Components

AgroManagement: Rule Based Modelling of Technical Management

The AgroManagement component was developed by CRA and is designed to implement field management actions during simulation. An agricultural activity is defined, in this context, as a production enterprise such as a crop rotation, i.e. an assemblage of crops, an orchard etc., associated with a production system characterized in terms of outputs and inputs such as high input, high output (e.g., irrigated, high nitrogen fertilization, minimum tillage). Such an integrated system must be implemented in a way that imitates as closely as possible farmers' behaviour. Limiting the drivers of the decision making process to the biophysical system implies that each action must be triggered at run time via a set of rules which can be based on the state of the system, on constraints of resource availability, or on the physical characteristics of the system. However, simulating management in a component-based system poses challenges in defining a re-usable framework which is able to account for the complete range of agricultural management technologies applied to particular enterprises. Finally, the implementation of management must allow different approaches to be used for modelling its impact on different model components.

The AgroManagement component formalizes the decision making process in models called rules, and the drivers of the implementation of the impact on the biophysical system as a set of parameters encapsulated in data-types called impacts. Rules and impacts are both easily extendable, thus allowing a wide range of modelling approaches to be used. Furthermore, the information on the biophysical system is passed through a data-type called states, which can also be extended in case new rules require additional variables. The outputs from the management actions, applied as a result of rules evaluated at run-time, needed to provide the simulation output (output to a text file, an XML file or a database are all currently available) can be fully customized by the user and added to without recompiling the component.

Currently, the management actions which can be implemented are nitrogen fertilization (mineral and organic), tillage, irrigation, pesticide application, and crop, tree, and grassland operations. The software implementation is such that new agro-management typologies, and new actions within the typologies, can be easily added.

The rule-based model is characterized by three main sections:

- Inputs: states of the system and time
- Parameters specific for each rule (values are compared to states of the system via the rule model)
- A model which returns a true/false output

Rules, which are based on relative date or on a set of state variables, are implemented as a class encapsulating their parameter declarations and tests of pre-conditions (this also allows management configuration files to be validated using pre-condition tests). One feature of interest is that implementing the rule approach allows the formalization of what is generically referred to as "expert knowledge".

For example, expert knowledge which suggests that in a specific environment, a farmer will "plant maize on a *date later than April first*, if it has *not rained for the last three days*, and when *average air temperature has been above five °C* for *seven days continuously*" can be formalized and used in simulations. The italicized words are the parameters of the rule to be compared with system states/exogenous variables at run-time (e.g. the condition "no rain for the last seven days" is tested against the values of rain at run time starting from April first as in this example). The possible uses of such formalization include building a consistent quantitative database of agricultural management across Europe, optimising parameters in climate change scenarios as an adaptation strategy and using such metrics in climate change impact assessments, and improving technical management in current conditions through rule-parameter optimization. Parameters are needed by model components to implement the impact of management actions. Some are common to many management events (e.g. management type) while others apply to a specific management event (e.g. amount of water for irrigation, tillage depth for tillage). Other parameters are needed by specific modelling approaches and generally differ even within specific management event types (e.g. implement type and an associated set of eight parameters is needed for modelling tillage according to the WEPP (Water Erosion Prediction Project) approach, Alberts et al. 1995, as opposed to other approaches which do not need such information). All model components reference the AgroManagement data-types to trigger management impact models at run-time. An example of the graphical representation of a management configuration for a 3-year rotation is shown in Fig. 4.3.

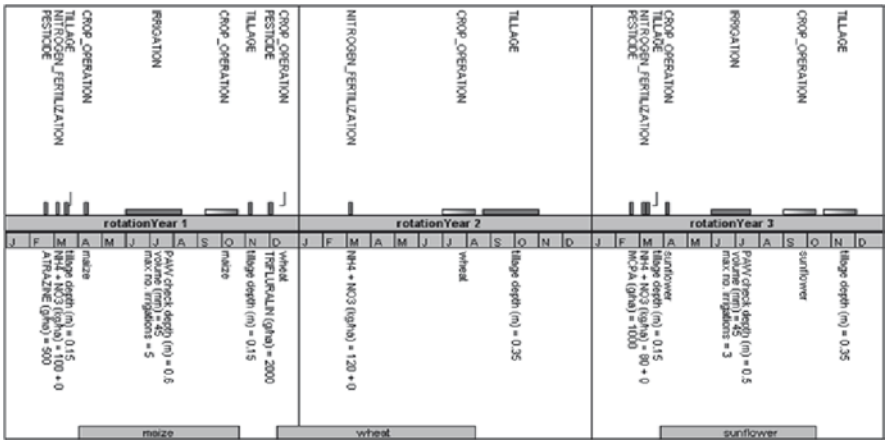

Fig. 4.3 Agro-management scheduled actions in a 3 year rotation. For simulations longer than this, the sequence is repeated. Vertical bars in the upper section of the graph are actions scheduled at a relative (to year) date; horizontal bars are actions scheduled in a time window, if other conditions are met; horizontal bars with a shading gradient are actions scheduled with an end date but associated with a phenological event (the width of gradient boxes is arbitrarily fixed as 30 days in this graphical representation)

Production Enterprise Components

AbioticDamage: Damage on Plants by Abiotic Factors

The AbioticDamage component was developed jointly by JRC and CRA. It implements several approaches for the simulation of abiotic damage to crops. Models are implemented with a fine granularity. The constituent models currently belong to five categories: lodging, frost, cold-induced spikelet sterility, ozone, and salinity.

- Lodging implements the approach proposed by Baker et al. (1998), modified by Acutis et al. (2008), assuming that the dominant parameter that affects lodging is the wind-induced bending moment at the stem base.
- Frost (Ritchie 1991) calculates crown temperature, hardening and de-hardening index, a killing temperature, the possible reduction in leaf area index, and evaluates if the crop has been killed by the frost.
- SpikeletSterility implements two different approaches, proposed by Confalonieri et al. (2006) and Shimono et al. (2005). An option model allows an automatic choice between them according to input availability. The Confalonieri approach is based on the computation of hourly stresses which are summed to compute the daily stress. The Shimono approach computes daily stress directly, but it requires the calibration of empirical parameters. The different susceptibility to sterility in the period between spikelet initiation and heading is accounted for by both models.
- Ozone contains a complex model for the simulation of the damage due to ozone. It models leaf aerodynamic and boundary layer resistance (Spiker et al. 1992), calculates average leaf conductance using the method of Georgiadis et al. (1995), and calculates the fractional reduction of plant production as a function of the ozone flux through the stomata and the leaf conductance of water using the approach of Sitch et al. (2007).
- Salinity implements two different approaches, proposed respectively by Ferrer-Alegre and Stockle (1999) and by Karlberg et al. (2006). The Ferrer-Alegre approach is based on the calculation of plant conductance and then of a function for the estimation of salinity stress in different layers of the vegetation. The Karlberg approach calculates the reduction of nutrients partitioned to the leaves due to salinity stress on the roots.

AgroChemicals: Pesticide Fate

The Agrochemicals component was developed by UNICATT, and is a one-dimension model that simulates the pesticide fate at field scale with a daily time step for communication with other modules; this component was developed by using new knowledge (Jantunen et al. 2005; Balderacchi et al. 2007) to modify earlier models (Carsel et al. 1988; Tiktak et al. 2001). The model considers five environmental compartments where the pesticide can be stored:

- Canopy surface: represents the pesticide deposit on branches, leaves, fruits, shoots and green parts (hence the outer part of the plant)
- Plant: represents the agrochemical stored inside the plant
- Soil available fraction: the pesticide quantity which can move and can be transformed by biotic processes
- Soil aged fraction: the pesticide trapped within soil micro pores and organic matter and not available for transformation
- Soil bound fraction: the pesticide fraction that cannot be extracted from the soil without altering its physical-chemical structure and therefore not available for transformation

The models were implemented in four modules, representing environmental compartments:

- Air considers the processes that happen before the product reaches the soil including drift and plant interception.
- Crop simulates only the plant mass balance, although in this first prototype plant is only a sink of pesticide.
- Canopy simulates the processes that happen on the leaf surface.
- Soil describes the pesticide flow through the soil profile. Each soil profile has to be split in numerical layers; the equations which describe the fate of pesticides differ for the top and bottom layers, because there are different boundary conditions.

The processes that redistribute the pesticide into the system connect two compartments and are:

- Penetration: from canopy surface to plant
- Wash off; from canopy surface to soil available fraction
- Ageing: from soil available fraction to soil aged fraction and vice versa
- Binding: from soil available fraction to soil bound fraction
- Plant uptake: from soil available fraction to plant
- Transport in liquid and gaseous phase: between the soil available fraction compartments of different layers

The output variables of greatest interest due to their importance for environmental pollution are: amount of pesticide lost due to drift; amount of pesticide volatilized, amount of pesticide lost due to run off, amount of pesticide lost to the drain system when present, amount of pesticide leached, and amount of pesticide remaining in the soil profile.

Crop: Crop Development and Growth

The crop component was developed by PPS and CIRAD. It simulates crop growth and development for the major crops of Europe. Tropical crops are currently being added. Crop growth is based on the interception of radiation by green plant parts and its conversion into dry matter. Crop development goes through vegetative

and, for some crops, reproductive phenological stages, at a rate that depends on physiological time, expressed as a temperature sum. The timing of readiness for harvest is also simulated. Potential crop growth is defined by temperature, radiation and crop phenology. The crop growth and development model simulates both potential growth and attainable growth, which is limited by water and nitrogen availability. The crop component uses a generic crop simulator in which parameters and modelling approaches can differ according to the crop simulated.

The crop component has been based on the concept of light interception and utilization from the Lintul model. However, modifications and additions have been introduced to extend the list of crop types for which the model can be used. These changes include the implementation of alternative modelling approaches for each of the main crop physiological processes, such as:

- Leaf area expansion
- Biomass accumulation (Monteith 1977)
- Biomass partitioning (van Keulen and Seligman 1987)
- Phenology (van Keulen and Seligman 1987; Streck et al. 2003; Hearn 1994)
- Senescence
- N dynamics (Shibu et al., 2009)

The model set-up allows new approaches for modelling these processes to be included easily. Parameter sets for 19 crops are currently available, including cereals, legumes, roots and tuber crops, and comprising determinate and indeterminate, and winter and spring crop types. The crop list can be extended not only by adding new crops but also varieties suitable for a particular environment, by editing existing parameter sets of the relevant crop type. In SEAMLESS-IF, these parameters are fine-tuned for regional applications by defining two correction factors that adjust crop cycle duration and crop radiation use efficiency.

CropML: Crop Development and Growth

The CropML (Crop Model Library) was developed by JRC, CRA, and UNIMI. The component implements alternative modelling solutions from those in the Crop component using different generic and crop-specific crop models. The architecture adopted allows easy extension of the component through the incorporation of other models. In fact, the fine granularity used for coding the different processes related to crop growth and development allow the re-use of the same strategies for other modelling approaches where common algorithms are present.

CropML is implemented as two separate components, CropML and CropML. Interfaces. The first includes all the algorithms (the models), the second interfaces, domain classes, and information about crop model parameters. Three versions of the component were developed: CropML, CropML.WaterLimited, and CropML. NitrogenLimited. The last two extend, respectively, CropML – CropML.Interfaces and CropML.WaterLimited – CropML.WaterLimited.Interfaces.

The models currently implemented are the plant growth and development approaches of CropSyst (Stockle et al. 2003), WOFOST (van Keulen and Wolf 1986), and WARM

(Confalonieri et al. 2006). The first two models are generic crop simulators, based respectively on the simulation of net and gross photosynthesis; the latter is a model specifically for rice simulations. CropSyst simulates net daily biomass accumulation using two approaches: the first is based on the concept of radiation use efficiency (RUE); the second is a vapour pressure – corrected Transpiration Use Efficiency approach. Each day, the minimum of the two biomasses is taken. CropSyst simulates daily biomass partitioning as a function of Specific Leaf Area at emergence, cumulated biomass, and an empiric parameter representing the partitioning of biomass between stems and leaves. WOFOST simulates the daily fixation of CO_2 (gross photosynthesis), the growth and maintenance respirations and a dynamic partitioning between leaves, stems and storage organs. WARM is based on the RUE approach, accounting for limitations to RUE due to temperature, senescence, saturation of the enzymatic chains and diseases. A dynamic approach for biomass partitioning of assimilates into stems, leaves and storage organs is also adopted in this case, driven by a single input parameter.

The CropML – WARM component uses a micrometeorological component, TRIS (Temperature in paddy-RIce Simulation). The TRIS component simulates the floodwater effect on the vertical soil thermal profile in paddy rice fields. TRIS is particularly important for rice simulations in temperate environments, where there is a significant effect of floodwater on temperature (one of the main driving variables in cropping systems models). Two alternative models were implemented, a mechanistic and an empirical one, for use according to data availability. The first is based on the solution of surface energy balance equations and estimates the temperature of floodwater, of each 10 cm canopy layer from the air–water interface to the top of the canopy, of the meristematic apex, and of the canopy. The model has an hourly time step. Context strategies allow also the generation or estimation of canopy and meteorological variables according to their availability in the domain classes. If needed, hourly inputs can be generated using the CLIMA libraries (Fig. 4.2). Maximum and minimum daily temperatures of floodwater, meristematic apex, and mid-canopy are calculated. The empirical model is based on modified Gaussian filters which reproduce the smoothing effect of water on daily thermal extremes, and the water heat storage capacity.

The component can be extended through the implementation of alternative approaches, e.g. for the simulation of meteorological variables into the canopy profile.

Diseases: Air-Borne Plant Diseases

The Diseases component was developed by CRA and UNICATT. It allows the impact of plant disease epidemics on plant growth and yield to be estimated. It consists of four modules providing a generic frame to simulate disease development:

- Disease progress
- Inoculum pressure (initial conditions)
- Impact on plants
- Agricultural management impact on pathogen populations

The Disease progress module simulates the epidemics of a generic air-borne fungal pathogen, considering the following components of the infection process: infection (Analytis 1977; Magarey et al. 2005), incubation, latency, infectiousness (Blaise and Gessler 1992; Wadia and Butler 1994), sporulation (Analytis 1977), and spores dispersal (Aylor 1982; Waggoner 1973; Waggoner and Horsfall 1969). These processes, which are driven by weather conditions and interactions with the host plant (Zadoks and Schein 1979), are modelled as a function of meteorological variables, temperature, air relative humidity, vapour pressure deficit, leaf wetness duration, rain, and wind speed – hourly values estimated/generated from the CLIMA libraries (Fig. 4.2), and parameters specific for each host-pathogen combination. As output, the Disease progress module returns the proportion of host tissue affected compared to the total host tissue.

The initial conditions for infections are derived from a pool of models which use information about the preceding crops, site-specific potential, and a random component obtained by sampling from a distribution, either provided by default or fitted from historical data. Impact on plants is currently implemented as a reduction of the photosynthesizing host tissue according to the Bastiaans' model (1991), but it will be extended to a more direct interaction with plant simulation as some air-borne pathogens such as rusts inhibit the conversion of solar radiation to dry matter. Finally, agro-management is accounted for through the impact of fungicide applications and other disease control actions on the fungal population. Prototype study applications have been made for vineyard diseases and powdery mildew of wheat (*Blumeria graminis* f.sp. *tritici*). The component also contains a model to simulate the pathogen rice blast (*Pyricularia orize*) and its impact on the rice crop, and a generic model for potential infection (Magarey et al. 2005), with parameters for more than 80 diseases.

Grasses: Grassland Growth and Quality

The grassland component was developed by INRA. It allows a wide diversity of grassland types to be simulated: (i) sown grass species including tall fescue (*Festuca arundinacea*), perennial ryegrass (*Lolium perenne*) and cocksfoot (*Dactylis glomerata*), sown legumes such as alfalfa (*Medicago sativa*), permanent grasslands ranging from plant communities growing under nutrient-poor to those from nutrient-rich conditions, and mixtures of grasses and alfalfa or white clover (*Trifolium repens*). For species-rich permanent grasslands the approach is based on plant functional traits (Lavorel and Garnier 2002; Duru et al. 2009).

The grass growth module is similar to that of the crops component except that additional functionalities are included:

- The calculation of the herbage feeding value: (i) protein content using the standing herbage mass and the crop nitrogen index; (ii) digestibility from herbage age, nitrogen index and plant type (Duru 2008; Duru et al. 2008).
- A detailed phenological sub-model (McCall and Bishop-Hurley 2003), for which parameters are specific to vegetation type in order to simulate a large range of defoliation regimes (cutting, grazing, short and long regrowth periods), over the vegetative or the reproductive phase.

In comparison to crops, some simplifications have been made:

- Leaf area is calculated only as a function of environmental factors and species type and not of dry matter accumulation.
- Over the reproductive phase, stem biomass accumulation is calculated as a fraction of the above-ground herbage mass (Calvière and Duru 1999).
- The root mass is considered constant over the growing season (as in many herbage growth models, e.g. Schapendonk et al. 1998; Barrett et al. 2005); i.e. root growth rate is assumed to be the same as the root death rate.

In grass-legume mixtures, the competition between the two components is not simulated. An a priori sward composition is defined that depends on the nitrogen and defoliation management.

FieldManager: Spatial Information for Multiple Plant Species

The FieldManager component, which was developed by INRA, provides field dimension information for heterogeneous stands such as agroforestry plots or vineyards where rows of trees or other woody perennials separate cultivated strips of either crop or grassland. Field configuration is defined by three parameters: WidthIntraRow (distance between trees in a row), WidthInterRow (distance between rows), WidthCultivatedStrip (width of cultivated strip between each row). Three field configurations are currently available: for crop alone, for a continuous row of grapevine with a cultivated strip of Crop or Grassland, for a row of timber trees with a cultivated strip of crop.

LightInterception: Light Interception and Competition by Canopies

The LightInterception component, which was developed by INRA, implements models that estimate the daily interception of solar radiation partitioned between one or two types of plant and the soil. From the daily solar radiation, a simple field description (from FieldManager) and a small number of plant parameters that characterize canopy dimensions (LAI, crown dimension), this component estimates the fraction and amount of PAR (Photosynthetic Active Radiation) intercepted by the tree canopy, any crop or grassland canopy below, and the soil. Three models are available in the current version, all derived from the geometrical model of Pronk et al. (2003), but corresponding to different field configurations (continuous rows, rows of cubes, or rows of cuboids). These models assume homogeneity in the light transmission properties of leaves and uniform canopies (this assumption is needed to permit the one-dimension simplification of canopy structure in other parts of the model). In the case of a single crop the model follows the Beer-Lambert law (Monteith and Unsworth 1990) for light interception.

MassBalance: A Library of Mass Balance Tests

The MassBalance component was developed by CRA. It implements equations of balance in discrete code units (strategies). The water and the nitrogen balance are currently implemented. At each time step the component computes the active balances, and outputs the values. Each balance has a threshold parameter as maximum departure from zero allowed; this threshold can activate a flag at run time. The component, which can be extended, is designed to be used either in the test phase of a new modelling solution, or permanently as a check on correct functioning.

RootDistribution: Roots Growth and Distribution

The RootDistribution component, which was developed by INRA, estimates the partitioning of fine roots between layers in the soil profile. Only one model is currently available, derived from Hi-sAFe 3D model (Root Voxel Automaton, Mulia 2005; Mulia and Dupraz 2006). From the daily fine root growth and death, the soil layer thickness and the water extraction of the previous day, the allocation of root length and biomass to the different soil layers is estimated using an opportunistic growth paradigm. Parameters can be fitted to adjust the water and distance to collar sensitivity so that root profiles of most species can be simulated. It is assumed that the horizontal distribution of crop roots is homogeneous, given that the root distribution is a one-dimensional simplification of the system.

Tree: Woody Plant Growth and Quality

The Tree component was developed by INRA. It is a generic woody plant model designed to simulate grapevines, fruit trees and timber trees; it is currently parameterized for grapevines. It simulates crop growth, production and product quality. Its basic features are similar to those implemented in the Crop component: temperature drives crop development, and intercepted radiation determines the potential growth rate that can be reduced by water or nitrogen limitations. Dry matter and nitrogen are partitioned between the growing organs on the basis of partitioning tables (Nendel and Kersebaum 2004; Vivin et al. 2002; Wermelinger and Koblet 1990). Some features are specific to woody crops. Where these crops are planted in rows, as is generally the case, their canopy is heterogeneous and the Tree component converts the daily carbon increment into an increase in crown dimensions (needed in the Light interception component, Pronk et al. 2003). Woody crops are perennial so carbon and nitrogen can be stored during the annual crop cycle and used during the next cycle (particularly at bud break) (Castelan-Estrada 2001). The Tree component calculates variables that define fruit crop quality, which is important for crops such as grapevines or apple trees. For grapevines, the dynamics of fruit water and sugar contents depend on the thermal time during the phase of grape growth after the onset of ripening (veraison) (Ollat et al. 2002).

The tree component does not use fixed partitioning tables for timber trees because this would prevent the modelling of the impact of branch or root pruning on the tree growth. The tree component includes a dynamic module of C allocation for timber trees, governed by two types of rule:

– Teleonomic allocation rules based on allometric equations define the relative sizes of above-ground sub-compartments, e.g. the relationship between Diameter at Breast Height (DBH) and tree height. Such allometric relationships capture internal constraints not explicitly dealt with in the model (e.g. architectural model and structural stability constraints limiting amount of leaf biomass for a given amount of wood biomass) in relation to the tree dimensions.
– An optimal allocation assumption ('functional balance') between above- and below-ground biomass mediated through stress indices, which assume that a plant allocates its biomass so as to maximise its growth rate under the given environmental conditions. This approach has been extended to the ratio between coarse and fine roots, with a dynamic allocation procedure that avoids the need for fixed partitioning tables.

WaterUptake: Plant Water Uptake

The WaterUptake component was developed by INRA. It allows daily plant water extraction to be estimated from each of the soil layers and for each plant species (one or two possible) in the field from the plant water demand, the soil description, and the root distribution in the soil. Roots are assumed to be homogeneously distributed horizontally. Two types of model have been implemented. The simple model for cases with up to two plant species, assumes that the water demand is met as long as water is available in the soil layers containing roots (water content above wilting point). The second model is the more complex one used in the His-AFe model for mixed vegetation. This model estimates the amount of water that each plant can extract from the soil by integrating the matric flux potential for each plant in each rooted layer.

Soil Components

SoilCarbonNitrogen: Soil Carbon and Nitrogen Dynamics

The SoilCarbonNitrogen component was developed by CIRAD and INRA. The models implemented describe N mineralization-immobilization turnover and the interactions between C and N dynamics in decomposing plant residues and soil organic matter (SOM). It includes above- and below-ground plant residue pools and three soil organic matter pools (microbial biomass, young and old SOM) with different turnover times (Fig. 4.4). Rates of decomposition are modified by temperature, moisture, lignin content of the residues and N availability. Stabilization of SOM is simulated by transferring fractions of decomposed microbial biomass and young

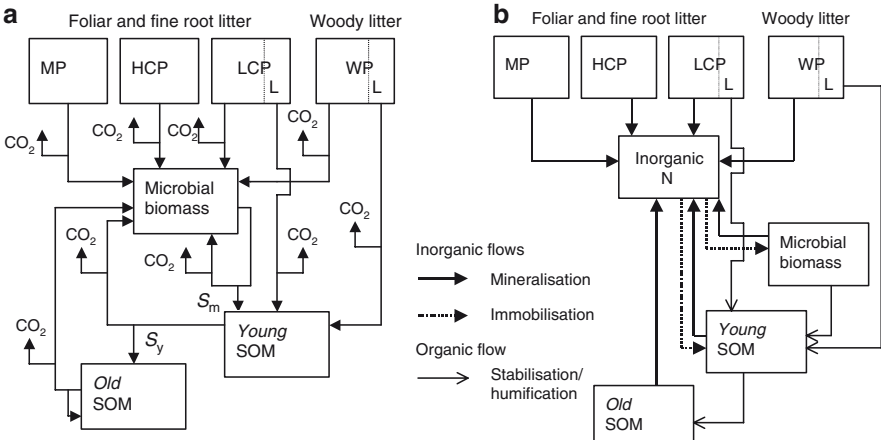

Fig. 4.4 Pools and fluxes of (**a**) carbon and (**b**) nitrogen in the new decomposition model. MP: metabolic pool; HCP: holocellulosic pool; LCP: ligno-cellulosic pool; L: lignin; SOM: soil organic matter; Sm: stabilisation coefficient for microbial biomass; Sy: stabilisation coefficient for young SOM (From Corbeels et al. 2005)

SOM into more recalcitrant forms (respectively into young and old SOM). Nitrogen is mineralized to, or immobilized from, the soil inorganic N pool to maintain the N:C ratio of decomposing microbial biomass within a specified range. Balancing potential microbial N demand against inorganic N availability determines whether the activity of decomposers is limited by N. If so, then simulated decomposition fluxes are reduced. The maximum rate of microbial N uptake is proportional to soil inorganic N content. Lignin incorporation in the young SOM pool results in additional N immobilization in the young SOM pool, which simulates the process of chemical N immobilization.

SoilErosion: Water Runoff and Soil Erosion by Water

The runoff-erosion component was developed by UNIMI. It simulates surface runoff and erosion, and handles irrigation events. The same soil description as in the SoilW component can be used. Runoff and erosion can be simulated daily when only daily rainfall is available, or for shorter time periods, if hourly or more frequent data are available. The following four processes are simulated, allowing for easy interchangeability and extension of options:

- Interception of rain by vegetation; two approaches (von Hoyningen-Huene 1981; Brisson et al. 1988) are available, both of which calculate interception as a function of Leaf Area Index.
- Interception of rain by mulch.
- Runoff using either the Curve Number approach, which is suitable for daily rain data (SCS 1972) or the kinematic wave approach, when hourly or more detailed

data are available. Infiltration of water in soil is simulated using either Smith and Parlange (1978) equation or Green and Ampt (1914) equation. Peak runoff is estimated with an empirical equation from EPIC (Williams et al. 1989), or is intrinsic in the Kinematic Wave approach.

- Erosion is estimated using the MUSLE (Modified Universal Soil Loss Equation, Williams and Berndt 1977) which is suitable for single rain events. The EUROSEM (Morgan et al. 1998) and Kineros (Woolhiser et al. 1990) models use a differential dynamic balance between splash erosion, shear stress of runoff water, carrying capacity of runoff water and deposition. RUSLE (Revised Universal Soil Loss Equation) using the adaptation of Cooley (1980) for single storm events is currently being developed for SoilErosion.

The SoilErosion component can also be used to simulate small hydrological basins because each simulation unit can accept as input runoff from an adjacent unit of simulation and can be either a plane or a channel.

SoilNitrogen: Soil Nitrogen Dynamics

This component was developed by UNIMI and is an implementation of SOILN (Johnsson et al. 1987), simulates the transformation of organic carbon and of organic and inorganic nitrogen in the soil. The model uses three pools to represent organic C and N: one is slow cycling (humus), and two are labile (litter and manure). Dead roots and incorporated crop residues are added to the litter pool, while animal faeces are added to the manure pool. Each input of organic matter is characterised by a specific N:C ratio and humification and ammonification coefficients, and is assigned to a "litter" or "manure" category. Inorganic N is represented by two pools, ammonium and nitrate. All transformations of C and N (except denitrification) are simulated with first-order kinetics, using environmental controls (soil temperature and water content) to modify decomposition rate constants. Denitrification is simulated with a zero-order kinetic. Potential decomposition of organic matter is simulated by calculating C flows from litter and manure to humus and from all pools to CO_2. Soil microbial biomass is implicitly represented as part of the two labile pools, which therefore represent the association of added organic materials with their decomposers. The following sources and sinks of ammonium and nitrate are simulated by the component: urea hydrolysis, nitrification, denitrification, atmospheric deposition, nitrate leaching, crop uptake, and ammonia volatilisation. Although denitrification and ammonia volatilization are implemented following the strategy pattern, there are no alternative approaches currently available.

SoilTemperature: Simulation of Temperature in the Soil Profile

The soil temperature component was developed by UNIMI. It allows soil temperature to be simulated down a one dimensional profile. The following processes are simulated:

– Surface temperature: The empirical approach of Parton (2004) and the mechanistic approach of Campbell (1985) based on the energy balance at the soil surface are both available.
– Transmission of heat in the profile: The Campbell (1985) approach based on a one-dimensional differential equation of heat transfer is available. The empirical approach of SWAT (Neitsch et al. 2002) is being developed.

SoilReader: Accessing Soil Data at Initialization

The SoilReader component was developed by CRA and UNIMI. It has four functions:

1. To load data (soil parameters, soil initial conditions, water table presence)
2. To estimate parameters which are either missing or which need to be estimated using pedo-transfer functions
3. To create soil layering from soil horizon data
4. To create daily values of water table depth

The component uses the PedoTransferFunctions (PTF) component (Fig. 4.2) to make estimates both of soil hydrological properties and of soil parameters needed by soil water retention curve models from the available soil information. The PTF component is implemented using the same design as other APES dynamic components.

SoilWater: Soil Water and Hydrologic Characteristics Dynamics

The Soil water component was developed by UNIMI (Acutis et al. 2007). It allows one dimensional water redistribution in the soil to be simulated, and the changes in soil physical characteristic after a soil tillage operation. A soil profile is represented as a series of superimposed horizontal layers. For each layer, hydrological properties are provided by specifying the parameters of the appropriate hydraulic functions. Alternatively, the HYPRESS pedotransfer functions (Wosten et al. 1999) can be used to calculate hydrologic parameters from soil texture, bulk density, and SOM, or the PTF component (see SoilReader) that includes a large collection of Pedotransfer functions can be used to provide estimates. In addition, it is possible to provide the soil water contents corresponding to field capacity and wilting point. When, for numerical reasons (i.e. a finite difference approach for water dynamics simulation), a finer soil layer definition is needed, a method is available to split existing pedological horizons into thinner soil layers. The following processes are simulated, allowing for selection of alternate approaches:

– Soil water distribution: three approaches are available, an empirical cascading model, a cascading model with travel time taken allowing for water contents greater than field capacity but preserving the speed of calculation of the cascading method itself, and a finite difference solution of the Richards' equation.
– Water evaporation: two approaches, CropSyst (Stockle et al. 2003) and Ritchie (1972) have been adopted.

– Water uptake by roots: five approaches have been used, CropSyst (Stockle et al. 2003), Ceres (Ritchie and Otter 1985), Swap (van Dam et al. 1997), EPIC (Williams et al. 1989), and "control" where water uptake is an external input, as a total amount or by layers. "Control" only checks if the external water requirement is consistent with the actual soil water content.
– Tillage: the WEPP approach (Alberts et al. 1995) is implemented.

The tillage model simulates the effect of tillage and the successive soil settling using a database of over 80 implements. According to the characteristics of the different tillage implements, SoilW reacts to the "Tillage" event by: (i) redistributing soil particles if layers with different textural composition are included in the tillage depth; (ii) redistributing organic matter; (iii) burying crop residues; (iv) redistributing water; (v) calculating a new soil bulk density and consequent changes in layer thickness; (vi) changing the retention and conductivity functions, and field capacity and wilting point; (vii) simulating the time course of soil settling with its effect on soil characteristics according to the amount of rain and time elapsed after the tillage event.

SoilWater 2: Soil Water and Hydrologic Characteristics Dynamics

The SoilWater2 component was developed by CIRAD and LIRMM. It mainly differs from SoilWater by taking account of preferential water flow through the soil profile. The model considers three structural levels within the soil profile, the pedostructure, the primary peds and the primary particles (Braudeau and Mohtar 2009). The clayey plasma of the primary peds, micro-porosity, and the inter-ped pore space, macro-porosity, are represented by two compartments which are in contact through a transitional zone at the surface of the primary peds. Water is distributed between these two compartments whose volume varies with water content. Water fluxes between compartments and from one pedostructure unit to the other result from an alteration in the hydrostatic equilibrium due to water supply (rain, irrigation) and evapotranspiration. Fluxes are simulated using Richards equations.

The functionality of the pedostructure is quantitatively described by equations that originate in the measurement of four soil characteristic curves: the shrinkage curve, the swelling curve (Braudeau and Mohtar 2006), the conductivity curve, and the soil water potential curve (Braudeau 2006; Martin et al. 2006; Braudeau and Mohtar 2009). Those four equations are described using 15 parameters which can be estimated using specific pedotransfer functions gathered by Saxton and Rawls (2006) and available in the PTF component, thus allowing model runs to be carried out with the same soil information used by conventional soil-water models. The SoilWater 2 component simulates the dynamics and interactions of soil structure and soil water. The profile consists of a surface layer and underlying horizons. The impacts of technical practices like tillage, or the effect of a soil surface crust, are on water infiltration and evaporation. Surface hydraulic conductivity, layer thickness and maximum surface storage are the three principal factors that are modified.

Each horizon has a pedo-structure, a homogeneous zone in terms of structure and organization of particles. The soil is divided into homogeneous layers.

The equations used allow the uniformity of the layer's depth in each horizon and the differences between horizons to be represented (Braudeau and Mohtar 2009).

Weather Data Components

WeatherReader: Accessing Weather Data and Estimating Missing Values

The WeatherReader component was developed by CRA. It has two functions: (1) to estimate missing weather data, and (2) to provide access at run time to location and weather data. The component is associated with components of an application provided as a separate tool (CLIMA), which allow missing weather variables to be estimated and weather to be generated from a rich library of alternative models implemented in six categories: AirTemperature, Evapotranspiration, LeafWetness, SolarRadiation, Rainfall, and Wind. The CLIMA application is described in the following paragraphs. Some estimation capabilities from the CLIMA components which are encapsulated in the WeatherReader and are active at run time, estimate weather data which are never available from weather data records (e.g. vapour pressure deficit, day length, extraterrestrial radiation)

The Intended Use of APES

Version 1.0 of APES allows rotations of crops and vineyards to be simulated for water- and nitrogen-limited conditions. The current modelling solutions allow one dimensional fluxes to be estimated. It is primarily a prototype for evaluating the adequacy of APES in terms of:

– The model framework, and in particular its ability to link operationally different model constructs within the simulation tool. Such an evaluation includes conceptual evaluation (criteria and needs for combining models implemented in discrete units), and a technical evaluation (adequacy of the modelling framework for linking components).
– The Graphical User Interfaces. This evaluation involves seeking feed-back from APES users on the stand-alone application. Users of the SEAMLESS integrated framework with a biophysical background are also able to test APES using a specialized user interface.

The system allows water, nitrogen, and pesticide dynamics to be simulated at the field scale in response to agro-management in the range of environments (soil-weather combinations) characteristic of the agricultural parts of Europe. The choice of spatial scale has been a direct consequence of the goals of the

simulation, namely to estimate production and system externalities in response to detailed agricultural management applied in specific soil-weather combinations. Modelling approaches selected and implemented in APES were mostly developed at field scale. Simulation outputs at this scale have been used in the literature to provide outputs at the regional scale by linking to Geographical Information Systems holding information on the spatial distribution of soils and weather. In such cases the most frequent recommendation is to use simulation outputs to make relative comparisons between different agro-management options. Other options are to use simulation outputs at the field scale as "cell" data to be integrated in spatially explicit models, as in some catchment models. In this case, the increased number of inputs needed generally limits the use of these models to case studies. All uses at scales other than the field scale involve additional assumptions that may be difficult to justify. Moving across scales is being addressed in SEAMLESS with specific actions, but it is outside the modelling domain of APES.
The optimum temporal scale is still a matter of debate, as opinions differ about the significance of possible drift in multi-year simulations without the re-initialization of state variables. However, this use is both a given and implicit in the simulation of multi-year crop rotations and is accepted in peer-reviewed publications on the use of tools like APES. In any case, the issue is not about APES itself, but about all model tools built with modelling approaches similar to APES.

As the simulation tool has been developed with a focus on modularity, APES versions including different modelling engines and components (modelling solutions) can be made available as "closed" modelling solutions to be used for situations where the assumptions made by their developers (modellers) apply. A set of options may be made accessible (e.g. to simulate reference evapotranspiration using different approaches), but in order to protect system integrity, APES users will not be able to access model composition (in their role of model users). However, APES is an open system so that the same individual, with a different role, may access model building, in this case taking the responsibility for the choices made. This is the expected use beyond the end of the SEAMLESS project. Simulations can be run using long series of either generated or observational weather data, to account for the stochastic variability of weather. Outputs can be evaluated as means and deriving measures of variability.

Inputs

Whenever alternative options are available to simulate a given process and such options perform almost equally well, the less demanding model in terms of parameters and inputs should be selected. However, data availability and quality cannot be allowed to limit the implementation of model capabilities when it prevents the achievement of the goals of SEAMLESS. APES releases minimize data requirements and use options, such as pedo-transfer functions and weather generators, to estimate missing variables and parameters from the available data.

The operational use of APES will highlight data gaps that need to be filled if it is to be used throughout the EU (both in absolute terms and in specific areas/countries). Such an analysis can be considered a side-product of the APES development action, but of specific importance. For instance, the formalization of agro-management using a common framework for current agricultural practices is a need which goes beyond its use in APES.

Current APES inputs can be grouped in six file types (the detail is provided in the documentation):

- Site data (e.g. latitude, elevation, …)
- Daily weather
- Soil data (e.g. clay, silt, and organic carbon percentages, horizon thickness) by soil horizon; slope, field length – unique values
- Soil initialization data (initial conditions for state variables; if missing, default values are used)
- Soil water table (if missing, the assumption is that no water table affects the root zone)
- Planned agro-management (see AgroManagement component)

Site and weather data are loaded at run-time from the WeatherReader component, which allows missing data to be estimated using CLIMA. Soil data are loaded at run time from the SoilReader component, which allows missing data and hydraulic parameters to be estimated using the PedoTransferFunctions component. Agro-management has proved to be the most challenging problem, because the lack of a common formalism to store information beyond its use in agricultural statistics (i.e. a static, summary description) makes it difficult to develop rules to simulate the dynamic part of farmers' decision making processes based on bio-physical drivers. APES development, and specifically the AgroManagement component has, however, provided a framework to formalize such data, making them of use for simulation of current and alternative agricultural management at field level.

Parameters

Parameters are defined as quantities that do not change value during either the whole simulation or parts of it. For example, crop parameters change when a crop is changed during the simulation, but their values do not change during the time a given crop is simulated. A simulation system which allows the use of more than one modelling approach cannot define a constant set of parameters for two reasons. First, some simulation approaches may need to model a parameter which then becomes a variable. For instance, a simulation system which does not model impact of tillage on soil physical properties will probably consider soil bulk density as a parameter; whereas bulk density will be modelled as a variable if the goal of simulation is to estimate the impact of tillage on soil hydrological characteristics. Secondly,

modelling approaches may differ in the number and type of parameters required. Model components encapsulate the knowledge of their own parameters and handle them in response to either initialization or agro-management events. Sets of parameters can be edited via a dynamic, generic parameter editor (see the paragraph about the application Model Parameter Editor). Default values are part of the definition of the parameter together with minimum and maximum values, as is the case for variables.

Software Architecture

One critical issue in model linking to assemble model components into a composite model, is the difficulty of finding a component design that satisfies the requirement of 'third-party composition'. In order to integrate my work with yours, *my component* must be compatible with *your component*, but frequently this is not the case: components are designed for a specific architecture or framework, and they are not usable outside it. Component design choices, rather than being peculiar to a specific architecture, should promote re-usability by including design traits which represent a compromise between level re-usability and complexity of the design chosen to maximize adaptability of components. Using a pragmatic approach, simplification can be obtained if the target use of components is within a specific knowledge domain. This has an impact not only on simplifying the design of components, but on clearly defining the scope of the knowledge domain which is embedded in the modelling exercise. Yet, restricting to a particular knowledge domain has often also meant restricting to a particular framework, where implementations of model components strongly depend on the modelling framework core. Targeting model component design to match a specific interface requested by a modelling framework decreases its re-usability. This explains why modelling frameworks, although in theory a great advance with respect to traditional model code development, are rarely adopted by groups other than the ones developing them (Rizzoli et al. 2005). One way to overcome this problem is to adopt a component design which targets intrinsic re-usability and interchangeability of model components (e.g. Carlini et al., 2006; Donatelli et al., 2009; Donatelli and Rizzoli 2008). Such components can be used in a specific modelling framework by encapsulating them using dedicated classes called "adapters". Such classes act as bridges between the framework and the component interface. The disadvantage of this solution is the creation of another "layer" in the implementation, which adds to the already implemented machinery in the framework. However, if components are correctly designed, there is a negligible, if any, penalty in performance, and the adapter does not add complexity. A further argument in favour of framework-independent components is that they allow modelling knowledge to be shared in a form which makes it easily re-usable.

Model components developed in SEAMLESS for use in APES are based on the above design paradigm: framework-independent components which can be linked to different modelling frameworks. A proof-of-concept of this claim has been

shown by Argent and Rizzoli (2004). MODCOM (Hillyer et al. 2003) is the linking framework used in the current release of APES.

Component Design

Components are discrete software units used for composition. Hence, components cannot be used in isolation. Further, the reason for adopting a component-based paradigm for implementing models as computer programs is to achieve specific functionalities not available with monolithic structures. Consequently, the component architecture and its implementation are crucial in developing a component based system for biophysical simulations that enables solutions to biophysical modelling problems to be implemented using different designs and technologies. Developing a design for a component architecture and selecting a technology for its implementation should be the result of a careful definition of requirements. The following requirements were defined for the APES model components:

Functional requirements:

- Estimation and generation of variables from different models
- Estimation of parameters from observational data
- Provision of data at run time, accessing either observational or generated data, and making available model outputs
- Provision of quality checks on imported data
- Provision of quality checks on outputs
- Robust behaviour of the component that degrades gracefully, raising appropriate exceptions
- Traceable component behaviour with traces that are scaleable, i.e. browsable at different debug levels

Non-functional requirements:

- Ease of use: the components must be easily usable by clients; this has an impact on technology and on documentation.
- Extensibility: the capability to easily add alternative processing capabilities to the ones of the component from the side of the component user, without needing to recompile the component, and using the same interface.
- Re-usability: the practical possibility of using the component in different software systems; ease of use and solution to a common modelling problem are the pre-requisites.
- Replaceability: the capability for components to be replaced by a different component conforming to the same specification. “Different” here means either a newer version of the same component, or an implementation from a different party.
- Availability of fit-for-purpose documentation of models, software design, and code.
- Successful unit tests: unit tests for each public method, input-output tests should be reported in the documentation.

Technological requirements:

– The component software implementation must be made using technologies with a widespread adoption.

There are several ways of matching the above requirements via software design and implementation. In the next sections we list the major choices made for the design and implementation of static and dynamic components developed for APES.

Ontology

A pre-condition of the successful integration and re-use of an existing component is that the modeller understands the meaning of the data elaborated by the model. Understanding means associating a variable with a shared concept, knowing its spatio-temporal extent, its dimension and units and so on. In order to facilitate this, the components must contain information, possibly extracted from a public ontology, which describes the variables/parameters used, and allows checks on input-output links and data quality tests at run time. Information consists of concepts (variables in this case, which can be seen as instances of the concepts) and of attributes for each variable, encapsulated using the VarInfo type implemented in a utility component. VarInfo attributes are: Name, Description, Minimum/Maximum/Default value, Units (Athanasiadis et al. 2006). The properties with respect to data flow are not included among these attributes as they are not an intrinsic attribute of the variable. In other words, a variable can be an input to one model, and an output from another.

This information from VarInfo is used in the domain classes described below. The components also contain internal information about parameters and variables, using the same VarInfo type. Such information is defined in the component and used as described in the section on pre- and post-conditions. Collections of variables that are associated with particular domains define the Domain Classes (Del Furia et al. 1995). For instance, we could define *SoilStates* as the collection of all measurements relevant to a specific goal chosen for soil modelling. Such collections can be manually entered by a user or automatically created, using the built-in reasoning features of the ontology. The definition of domain classes in the component interface allows the dependency of the model to be abstracted from the data and the extensibility of models fostered using design patterns (Mesketer 2004; Bishop 2008). The importance of domain classes goes beyond their meaning as software implementation items. In fact they provide a detailed description of the domain of interest. Using domain classes, a modeller can exploit the knowledge structured in the ontology in different modelling frameworks and programming languages. The adoption of an ontology-driven approach for defining a model interface has clear advantages as it enables the reusability of models in an easier way, while common problems related to poor semantics of model interfaces can be effectively tackled.

The APES ontology is browsable on the web at a dedicated page (see Web resources) in which each domain class and strategy is described using the VarInfo values of their variables and parameters.

Model Granularity in Components

One definition of a model is that it is a conceptualization of a process. From a software point of view a model can be implemented in a class that offers a method for computing a variable (or a set of interrelated variables), thus obtaining the desired level of granularity. There might be more than one way to do this. If two different models estimate the same variable A, we can implement both of them as alternative methods for estimating variable A even if they have different input requirements and different parameters. To do this, the two models must be made available as separate units, and their input, parameters and output must be defined. Such units are here called "strategies", from the related design pattern introduced below.

A set of alternative models can be implemented in the same component using the design patterns Strategy and Composite. These designs offer the component user alternative algorithms (strategies) for doing the same thing. When building a biophysical model component, this allows in principle alternative options to be offered for estimating a variable or, more generally, for modelling a process. This often-needed feature in the implementation of biophysical models comes with two very welcome benefits on the software side: (1) it allows easier maintenance of the component, by facilitating the addition of other algorithms, (2) it allows the easy addition of further algorithms from the client side, without the need to recompile the component, while keeping the same interface and the same method signature.

In summary, the strategy (a model class) encapsulates a model, the ontology of its parameters and the test of its pre- and post-conditions. It can be used either directly as a strategy (in this case we call it "simple strategy", where simple indicates that does not use other strategies as part of its implementation), or it can be used as a unit of composition. A composite strategy differs from a simple strategy because it needs other (simple) strategies to provide its output(s). A sequence of calls might be implemented inside a composite class. The list of inputs includes all the inputs of all classes involved (except those which are matched internally). The list of outputs includes all outputs produced by each strategy and the ones specific to the composite class (if any). The list of parameters needed includes those of the classes associated with and the ones (if any) defined in the composite class. When the value of a parameter is set, if the parameter belongs to an associated class, it is set in that class. The test of pre- and post-conditions (see following paragraph) makes use of the methods available in each associated simple strategy class, plus the new tests specified in the composite class. If a violation of pre- and post-conditions occurs in one of the associated classes, the message informs the user not only about the violation that has occurred, but also in what class it has occurred. Composite strategies do not differ in their use compared to simple strategies.

A different type of composite strategy is the context strategy. Such classes implement an internal model to select the appropriate strategy (either simple or composite) based on the *context*, that is, on the inputs received at each call. An example of a context strategy is the one which estimates reference evapotranspiration by the Penman-Monteith, Priestley-Taylor, or Hargreaves method depending on the inputs available.

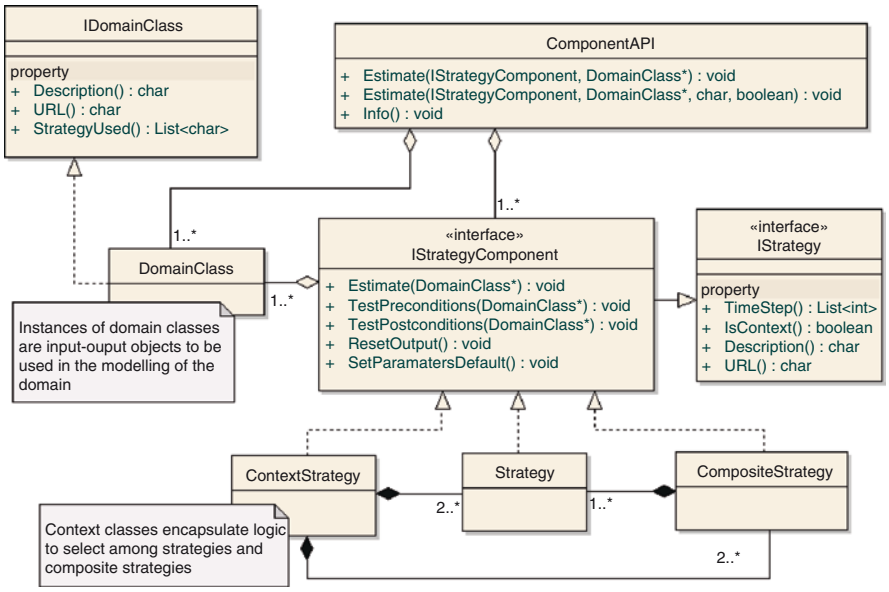

Fig. 4.5 Class diagram of a generic component according to the design used in APES components

The sample class diagram of a generic component in Fig. 4.5 shows the different type of strategies and the common interface which allows the design patterns described above to be implemented. The common interface IStrategyComponent and the capability to inherit from DomainClasses to extend them allows extending the component independently by third parties, and still using the original ComponentAPI.

The interface used for models is the same for all modelling solutions in the component and can be seen as an implementation of the Façade design pattern (Bishop 2008) to hide the complexity of model solutions based on composite strategies. This leads to their being a single signature for internal and extended models. An example of simple and composite strategies is given by Villa et al. (2006). Composite strategies too can be added to the components without requiring a recompilation of their code, thus providing a way to extend component models fully autonomously by third parties. Composite strategies are solutions to modelling problems at a coarser granularity (in principle) than that of simple strategies. As an example, a composite strategy may be built to simulate "crop potential production" and be developed by putting together simple strategies such as "light interception", "crop development", "leaf area expansion", etc. In other terms, a composite strategy is a "closed" solution which makes use of selected models of finer granularity as units of composition. Such a closed solution is not proposed as the unique solution for a specific modelling problem in a component as other options can co-exist or be added by third parties. Referring to the example above, two composite strategies may use different simple strategies to simulate "light interception" depending on whether they target the simulation of homogeneous canopies or wide-row crops. Even though such

differences in light interception models may not lead to noticeable differences in simulated potential yield, they may lead to sharp differences when simulating water-limited production, particularly in arid environments. Further, two alternative approaches to modelling light interception, say for "homogeneous canopies", could be implemented as two composite strategies, and this would allow modelling approaches to be compared at finer granularity. This kind of composite model provides a sound foundation for selecting modelling approaches to be used operationally.

Formalizing models in basic units of composition (simple strategies) and in aggregated units (composite strategies), with the same interface, and decoupling interfaces and data from modelling equations provides the design infrastructure to link and populate a knowledge base. The use of semantically rich interfaces fosters safe re-usability of components as discussed in the introduction. Finally, both simple and composite strategies are discrete units of code which can be used either to build components, or even as "full" simulation models to be used in stand-alone mode, in the latter case still preserving the benefits of a modular system as described in the introduction. By imposing the same interface on simple, composite and context strategies, the components obtain the full benefits of the Strategy and Composite design patterns, making re-use simpler and allowing full extensibility.

Test of Pre- and Post-conditions

Implementing tests of pre- and post-conditions is the central idea of the Design-by-Contract approach (Meyer 1991). In DBC software, entities have obligations to other entities based upon formalized rules between them. A functional specification, or 'contract', is created for each module. Program execution is then viewed as the interaction between the various modules bound by these contracts. In general, routines have explicit pre-conditions that the caller must satisfy before calling them, and explicit post-conditions that describe the conditions that the routine will guarantee to be true after the routine finishes. When building biophysical models, the DBC approach not only ensures the correct functionality of the software, but also specifies the limits of valid use of the model, which is knowledge about the model itself. It also allows data of uncertain quality to be used: if an input (either an exogenous variable or the output of another component) is out of the expected range, an exception can be raised, both informing the user of the problem and allowing for exception handling. The DBC approach is implemented in APES via a utility component developed for the purpose, called Preconditions.

Exception Handling

Exception handling is a programming language construct designed to handle runtime errors or other problems (exceptions) which occur during the execution of a computer program. Handling exceptions is of crucial importance in a component

based system as it allows users of the applications and subsystems using the components to know precisely the source of the error and thus to choose what to do in response, preventing hard-crashes of the whole application. Components raise and propagate exceptions and provide a customised message informing users which component and class are the source of the error. If an unhandled exception occurs, an informative message describes the error, and model and component source of the exception, allowing for continuing execution of the client if the user chooses.

Tracing

The traceability of component behaviour is implemented in.NET versions using the TraceSource class in an implementation that allows the client to set receivers of the messages called listeners. Various levels of tracing (critical, error, warning, information, and verbose) can be pooled in one or more listeners with all the traces from other components and from the client. Traceability is used in components to provide a log of execution shown at run-time in the APES stand-alone application.

Unit Tests

In computer programming, a unit test is a procedure used to verify that a particular module of source code is working properly. The principle underlying unit tests is to write test cases for all functions and methods so that whenever a change causes a deterioration, the cause can be quickly identified and fixed. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing provides a strict, written contract that the piece of code must satisfy. Beyond the general benefits which derive from unit test implementation in software development, implementing unit tests to test model implementation and make available the relevant input-outputs in the documentation allows the user of the components to have sample application results for the specific model.

Model and Software Design and User Documentation

Each component requires a help file which contains detailed documentation about the models implemented, and information about the design and use of the component. The test of documentation adequacy is that it should allow re-implementation of all the models that comprise the component, although the characteristics of re-usability of the component make it much easier to use it again rather than to duplicate it. The code of each component should also be thoroughly documented, so that

automated documentation processing utilities, such as those available for the .NET development environment, can generate the documentation reports.

Component Public Interface

One of the key elements for component adoption by third parties is simplicity of default usage cases via the Application Programming Interface (API). The usage model for component-oriented design follows a pattern of instantiating a type (a class) with a default or relatively simple constructor, setting some instance properties, and finally calling some simple instance method. This "Create-Set-Call" pattern (Cwalina and Abrams 2006) has been implemented in the APES components.

The component API contains one method for each of the strategies using the same signature (see the paragraph Model granularity in components). Each of the strategies uses the same signature. Domain classes and strategy inputs, parameters, and outputs can be found using the Model Component Explorer application described below.

Technology Used

The technology used is based on the Object Oriented Programming (OOP) paradigm as implemented in the Microsoft .NET 2.0 framework. However, the object model of.NET allows easy migration to the Sun Java platform. Such migration has been realized for some of the components referenced. Most of the components have been made available as discrete units inclusive of a software development kit with example projects in which to use the component by desktop clients, by web services and applications, so that components can be extended independently of their source code.

Model Component Diagram

Each of the model components represented in Fig. 4.2 is actually a package of discrete units. Figure 4.6 shows the component diagram of a generic APES model component.

If a component implements models to simulate agro-management, then it must have a dependency on the components CRA.AgroManagement and CRA. AgroManagement.Impacts. A component may have dependencies also on other components, such as numerical or statistical libraries, but must have no dependency on any modelling framework. Descriptions of model architecture are available in

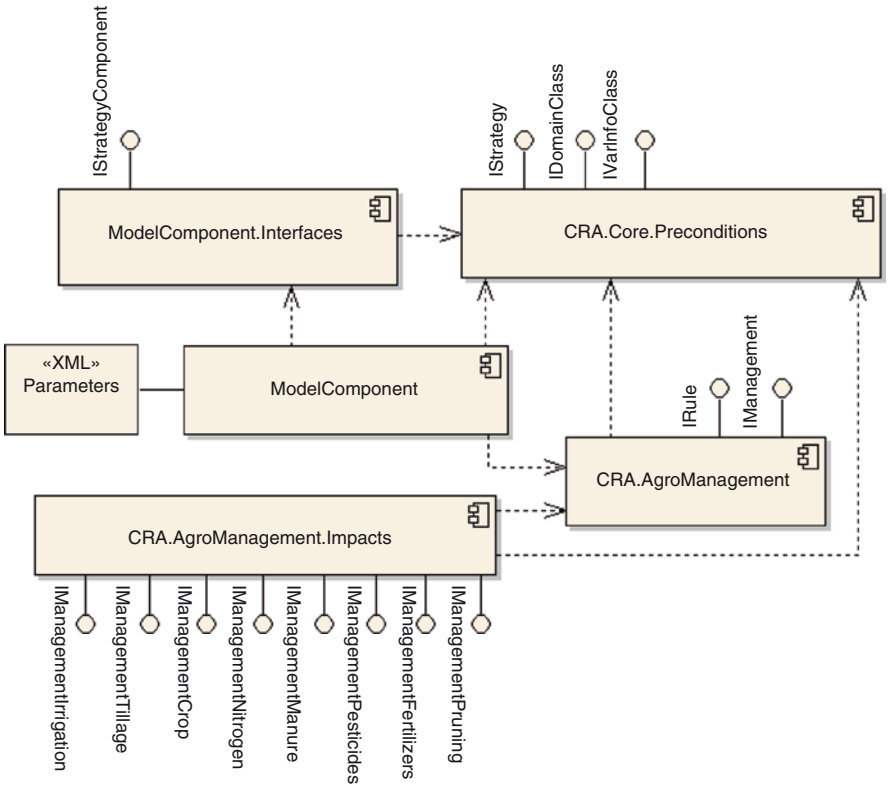

Fig. 4.6 Model component diagram. The MODCOM adapter component is not represented in the diagram (see Fig. 4.7)

the documentation of each component. Note that in APES, model components are encapsulated in an adapter class inheriting from a MODCOM class so that the MODCOM framework can be used, as described in the following section.

The MODCOM Engine

Components exchange data via the modelling-framework MODCOM (Van Evert and Lamaker 2007), which was developed by PRI. MODCOM is a software framework that facilitates the assembly of simulation models from previously and independently developed component models. It offers connectivity, time and state events, and numerical integration. APES components are registered via adapters to the MODCOM application which serves as the model engine, implementing the Adapter pattern as shown in Fig. 4.7.

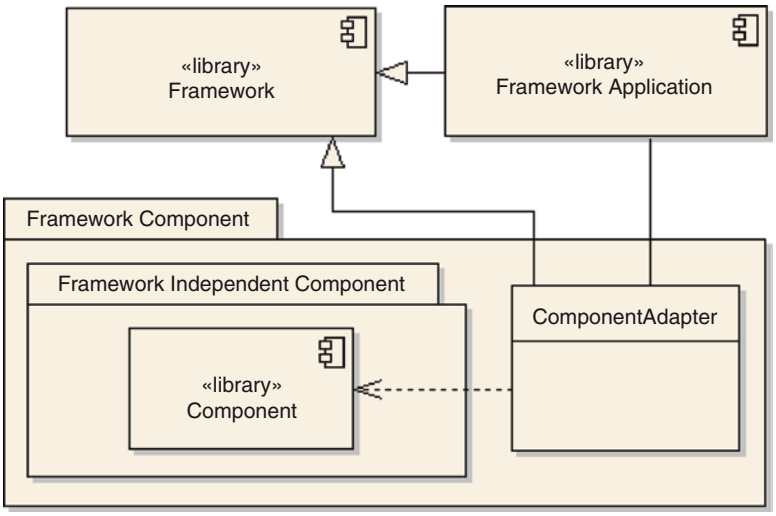

Fig. 4.7 Linkage of a generic component via the Adapter pattern in APES applications using MODCOM

Data Exchange Between Components

Data are exchanged at a time step of 1 day, but within each time step components can communicate up to three times. Within a time step, components can communicate, for instance to meet supply and demand computed by two different components, thus allowing estimation of actual rates besides the potential ones which do not need to match supply (e.g. for water and nitrogen). The multiple calls within the time step also permit arbitration of a source between two or more sinks. Details are provided in the MODCOM documentation. The fine granularity, strategy-based discretization of models has been shown to be adequate for accommodating multiple calls for different purposes within a time step, and allowing each component to be developed without any dependency on other components or on the modelling framework itself.

The APES Stand-Alone Application

The APES stand-alone application allows different instances of the modelling engine to be run using the following conventions:

– APES modelling solutions using alternative components (e.g. SoilWater or SoilWater2) are kept separate and can be loaded by the user.
– Once an APES modelling solution is loaded, a user interface page is dynamically built allowing modelling options to be set within components (e.g. potential growth or water limited growth options in the Crop component; the curve number or the kinematic wave approach for runoff).

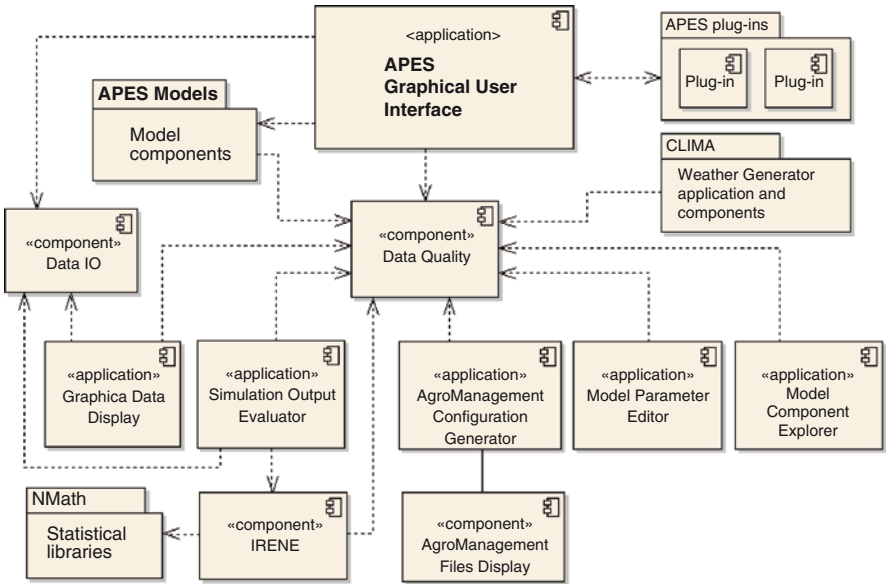

Fig. 4.8 The main components and packages of the APES stand-alone application. Connections represent dependencies. The package "APES Models" represents different realizations (modelling solutions) developed using the items of Fig. 4.2. All application tools (third row from the top) are reusable independently by third parties in custom developed applications. The component "Data Quality" is the component Preconditions cited in the paragraph Ontology

The APES stand-alone allows new applications to be added as supporting tools (plug-ins) independently by third parties. An example is provided inclusive of source code.

The application is also built using the component-oriented paradigm; single components can be re-used independently of APES. Figure 4.8 shows the main components of the December 2008 version.

The AgroManagement Configuration Generator

The AgroManagement Configuration Generator is an application developed by CRA to build XML files containing a set of planned agricultural management actions, and to visualize configurations. Such files can be used by the CRA. AgroManagement component at run-time to simulate the decision making process to implement agro-management actions in a field. As described in the documentation for the AgroManagement component, agro-management is simulated by parameterizing rules (classes implementing the interface IRule)

and testing them against the state of the system. If a rule is satisfied, the coupled impact (a class implementing the interface IManagement) is published. The ACG uses rules and impacts to build sets of rule-impact couples. ACG allows also files to be merged and uses the component CRA.AgroManagement.AFD to display planned agro-management actions using a graphical metaphor (see Fig. 4.3).

The Model Parameter Editor

Developing and maintaining a simulation system implies, among other things, that the parameters used can change. Composite models are made of simpler models, which can be often replaced by alternative formulations. This means that the development and management of a simulation system requires the ability to deal with changing the number and type of the parameters of the composite model when a sub-model is substituted. If the system consists of interchangeable components, the need for dealing with different sets of parameters is an inherent feature of the system; an alternative component may model the same domain variables, but its approaches may demand different, model-specific, parameters.

The need for changing parameters has a primary impact on the Graphical User Interfaces developed for the system: such user interfaces must be easily maintainable, and ideally present the same look and feel to the user when different sets of parameters are in use. Moreover, there should be a facility to check the accuracy of all parameter values. A parameter editor with these features must allow the parameters to be edited to be changed without changing the code, hence without a need for re-compilation of the editor.

The Model Parameter Editor (MPE) is an application developed by CRA (Di Guardo et al., 2007) which allows a dedicated user interface to be generated for each available parameter definition. It groups interfaces in different tabs either according to user criteria, or according to the model components which originate the parameter definitions. The application allows selection of parameter definitions, or it loads automatically parameter definitions from a folder of choice. MPE can be used as a stand-alone application, but it is meant to be used primarily in a simulation system like APES, becoming one object of the Graphical User Interface.

The Graphic Data Display Component

Providing data views from Graphical User Interfaces is a common need for applications built to make use of models. If model output is generated by a modular system in which model components are interchangeable, output variables may change. Thus, maintaining GUIs can be challenging and resource demanding. A tool which can load datasets with various schemas and which helps the user to visualize data in a range of ways speeds up application development, allowing the user to focus

on the models, rather than on the user interfaces. Whenever flexibility of use is important, providing domain specific views of data adds value to such a tool, both in operational usage and in model development.

The component GDD (Graphic Data Display) is a Microsoft .NET component developed by CRA (Di Guardo et al., 2007), which has the specific purpose of retrieving a set of output variables and allowing values to be displayed either as textual tables or as graphs. GDD can be used as a stand-alone tool or as a component inside an application. In the former case it provides access to a file dialogue that allows the user to select a file, whereas in the latter case it can be opened inside a modelling framework to directly load the current dataset. GDD accepts inputs in three different formats: XML, MS Excel, and the more compact and faster binary form (another available component also allows I/O operations with the binary format). Readers can however be extended by third parties implementing the proper interface. Each variable can be either a table column, or an entire table of the dataset, depending on whether it is either only time-variant or time and one-dimensional space-variant (the latter are variables that vary down soil profiles, such as soil temperature). GDD has seven tab pages supporting data views such as tabular views (which can be saved using the Microsoft Excel format), scatter graphs, time courses, histograms, soil profiles (water, temperature, nitrates, agrochemicals), 'Micale' graphics, frequency histograms, and probability of excedence. Also, GDD allows showing reference data against simulation outputs via configurations which can be saved. GDD can read APES GUI output files, in both XML and binary formats.

The Simulation Output Evaluator

The Simulation Output Evaluator (SOE) is a data analysis tool developed by CRA that provides easy access to model evaluation techniques. As the literature gives neither a standard theory on model evaluation, nor a standard "box of tools", the emphasis is on statistical techniques for comparing estimates either with actual measurements, or two series of estimates, making use of an extensible library called IRENE (the.NET 2.0 version). Non-replicated estimates are mostly compared with the non-replicated measurements. The program also allows comparison of individual estimates with replicated measurements (or vice versa) and replicated estimates with replicated measurements. The program provides extensive statistical capabilities with tools for a variety of needs. Ready-to-use procedures handle a wide range of statistical indices and test statistics. Basic statistics allow a preliminary check of data quality. The evaluation of model performance is based on either the model residuals or on the correlation coefficient. In addition, model evaluation by probability distributions (i.e., probability of excedence), residual analysis (i.e., pattern indices), and fuzzy-based aggregation statistics are allowed both for indices produced internally by the component and for external numerical values. The fuzzy aggregation model is saved as an XML file. Graphics are included in most analytical

tasks and the user can request many types of graphs directly. SOE can read APES GUI output files, both in XML and binary format.

The CLIMA Weather Generator

CLIMA is a Windows application developed by CRA (Donatelli et al. 2009b) which generates and estimates daily and hourly values of weather and weather-related variables using several alternative models (Carlini et al. 2006; Donatelli et al. 2006a and 2006b; Donatelli et al. 2009). It exports data in a format readable by APES, and custom writers can be added. CLIMA was built following the component-oriented paradigm and it is an example of re-use of components developed for APES. The component architecture is the same as other APES components, hence allowing both for component extensibility autonomously by third parties and for their re-use in custom developed applications (Fig. 4.9). CLIMA allows for composition of models into a composite model which is then saved as a discrete DLL. Such DLLs can be used either for data generation within CLIMA, or separately in a custom application.

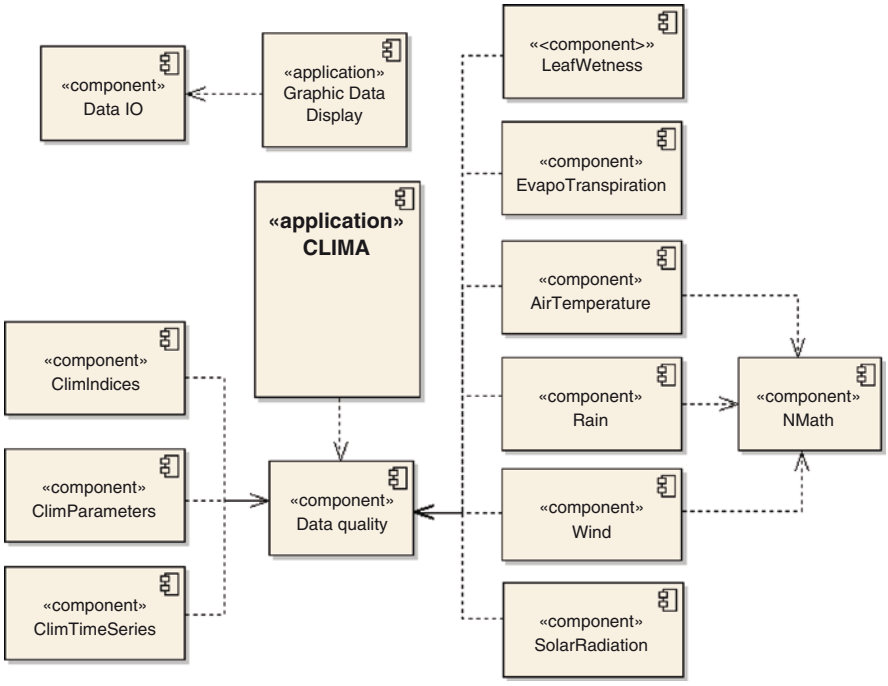

Fig. 4.9 CLIMA component diagram (main components shown)

The Model Component Explorer

The Model Component Explorer (MCE) is a Windows application developed by CRA for inspecting model components to reveal interfaces, domain classes and VarInfo values for each variable, simple and composite strategies and their parameters, inputs, outputs, and associated strategies. Given the component architecture implemented, the tool can be used to show the input and output variables within the domain classes of any model. The tool also allows XML files to be saved with schemas for each domain class and strategy, allowing them to be uploaded into a shared ontology (see paragraph Web resources).

APES Tools for Integration in Broader Modelling Systems

Given its component-based structure, APES can be run not only from a user interface, but also using a command line procedure. This allows the model to also be called from applications developed using languages which have no binary compatibility with .NET (e.g. Java) provided these applications are on a machine running Windows. APES is used in the software system SeamFrame (Wien et al., this volume) as an external component. As SeamFrame is implemented in Java, it requires APES to run as executable files. Support for integration can also be provided if the application which acts like a client is a web application, provided that such an application exposes web services and includes rich clients. In this case, some of the applications implementing a user interface can also be used. APES includes tools for integration.

The Parameter Estimator

Parameter estimation is a major aspect of crop modelling. Together with the functional forms of the equations, it is a major determinant of prediction quality. Parameter estimation is a difficult and time-consuming exercise and requires expertise not always available in a modelling project. The purpose of the Parameter Estimator (PE), developed by PRI and INRA, is to provide software to automate model parameter estimation. The Parameter Estimator consists of functions in the R statistical computing the language (R Development Core Team 2007). Models such as APES are coupled to the PE through specific functions. APES is written in the C# computer language and is, for the purposes of the PE, exposed as a Microsoft COM object. This makes it possible to use the R-to-COM bridge (Baier 2007) to execute an APES simulation run from within R statements. The R software requires the following information: the observed data, the name of the model to be run, the paths to the input files for the contexts of the data (for example climate, soil and management files for each context), the list of model parameters and indicators as to which are to be estimated and finally information related to the correspondence between the data and the model output. The R routine sets the parameter values of APES to the current values at each iteration, executes the model for each context,

retrieves the results, calculates the objective function to be minimized and determines the parameter values for the next iteration.

The current version of the PE has five different algorithms for parameter estimation. The different parameter estimation methods correspond to different hypotheses about model error.

The Sensitivity Analysis Component

Sensitivity analysis (SA) is a fundamental tool in the building, use and understanding of all types of mathematical model. SA provides information regarding the behaviour of the underlying simulated system. This information ranges from the identification of relevant model factors (parameters or inputs) to model reduction or simplification, better understanding of the model structure for given components of a system, model quality assurance, and model building in general. Among the most commonly used methods, it is possible to identify three classes: screening methods, regression-based methods, variance-based methods. The most used screening method is the one proposed by Morris, which is particularly effective in identifying the few important factors in models with many factors or with high computational requirements. The second class includes the regression methods, which are based on the computation of standard or partial regression coefficients quantifying the effects due to a change in a factor value while the others are kept constant. Within this class, different methods can be used to generate the sample of factors combinations necessary to obtain the model evaluations and therefore to calculate the regression coefficients; here, Latin Hypercube Sampling (LHS), Random, and Quasi-Random LpTau will be used. The last class, variance-based methods, includes the Fourier Amplitude Sensitivity Test (FAST), its evolution Extended FAST (E-FAST), and the method of Sobol'. All the methods belonging to this class compute total sensitivity indices for first and higher order effects and are demanding in terms of computational time because of the high number of model evaluations needed for each model factor. SensitivityAnalysis is a component developed by JRC and CRA (Donatelli et al., 2009c) with the goal of making available the sensitivity analysis models implemented in the SimLab library (Saltelli et al. 2004) via a user friendly application programming interface, in the memory managed environment of the Microsoft. NET platform (the Simlab library is available for C, C++, Matlab and Fortran). The component allows sensitivity analysis to be run on a model of choice using the methods mentioned above. It is implemented using C# under the Microsoft .NET v 3.0 platform. Sample applications inclusive of source code are provided to allow an easy start to SensitivityAnalysis use via different software clients.

Remarks on APES Integration in Larger Systems

APES integration, although technically possible at even closer levels than the ones used in the integration into the SEAMLESS integrated framework should, however, be approached with caution, providing users means to access and verify results of

any operation involving simulations, such as simulation *per se* or finalizing parameter calibration. This is not because of the component based structure of APES, but rather because complex systems are being simulated with process-based models. Automated optimization procedures in model chains may produce results which are meaningless in biophysical terms. Such simulation anomalies due to either inappropriate input data or even the misuse of the simulation model might be evident working with the biophysical system simulation alone, but a misuse of APES (or of any other process based system) in model chains may be very difficult to spot and could have an unpredictable impact on final results of the analysis. When included in a model chain, it is advisable to link APES simulations to other models asynchronously, allowing for simulation results to be evaluated by an analyst prior to using APES outputs as inputs for further processing.

The paragraph above is not meant to suggest that a complex simulation system should not be integrated in model chains such as the SEAMLESS integrated framework. Instead, it is meant to stress the importance of implementing procedures to facilitate the evaluation of intermediate results, both by domain experts and via specific utilities.

Concluding Remarks

APES development represents a paradigm shift for two reasons. First, APES is not proposed as "the" model. Instead it stresses the need for broadening modelling approaches and for comparing them at a finer granularity than for whole simulation systems. Secondly, compared to the first modelling frameworks for overcoming the problems of monolithic models, APES moves the focus onto components and their re-usability outside APES itself, even as stand-alone entities. A somewhat surprising result emerged during the initial development of APES. Contrary to past experience when implementation of complex systems has often been the most challenging task, the major difficulty has turned out to be thinking in "modular", "multiple choices", "transparent" modelling terms. The goal of making models available as discrete, re-usable units aimed at including ideally one process in each basic model unit has forced us to thoroughly analyze assumptions and the independence of each model from others. In fact, developing model components, even with the requirements listed in the previous paragraphs, is a modest challenge in terms of implementation, but it forces us to formalize modelling knowledge and the problem of model linkage and re-use. Technology is expected to move more and more towards declarative modelling in an operational way. The work carried out in creating fine-granularity, discrete model units, encapsulating a semantically-rich description of interfaces, has involved discussing and advancing understanding of many aspects of model assumptions and structures, and will be of great help in that direction.

APES development during the SEAMLESS project has led to an increasing opportunity to concentrate on modelling options by re-using expertise in different domains. APES is offered as a complete simulation tool, but also, and of no less

importance, as a loose collection of model objects which allow the modelling knowledge that APES teams have assembled to be shared operationally. Utilities and applications are also available as independent objects for re-use. A third party may want to use a single component or an extended set of them. They will be fully documented and extensible so they can be easily used in custom-developed applications.

Web Resources

The APES portal: http://www.apesimulator.org
Component and applications documentation pages: http://www.apesimulator.org/help/
APES ontology: http://www.apesimulator.org/OntologyBrowser.aspx
SEAMLESS EU integrated project: http://www.seamless-ip.org, partly continued in the SEAMLESS Association (www.seamlessassociation.org)

Acknowledgements The development of APES was partially funded by the EU – DG Research, Sixth Framework Research Programme, Integrated Project SEAMLESS (http://www.seamless-ip.org), contract no. 010036-2

References

Acutis, M., Confalonieri, R., Donatelli, M., & Rana, G. (2008). *Modellazione dell'allettamento dei cereali a paglia*. Proceedings of IX National Congress of Agrometeorology, pp. 84–85.

Acutis, M., Trevisiol, P., Gentile, A., Ditto, D., & Bechini, L. (2007). *Software components to simulate surface runoff, water, carbon, and nitrogen dynamics in the soil*. Proceedings of Farming Systems Design 2007, Catania, Italy, 10–12 September, 2007.

Alberts, E.E., Nearing, M.A., Weltz, M.A., Risse, L.M., Pierson, F.B., Zhang, X.C., Laflen, J.M., & Simanton, J.R. (1995). WEPP Model user guide. Chapter 7. *Soil component*.

Analytis, S. (1977). Über die Relation zwischen biologischer Entwicklung und Temperatur bei phytopathogenen Pilzen. *Phytopathologische Zeitschrift, 90*, 64–76.

Argent, R. M. (2004). An overview of model integration for environmental applications – components, frameworks and semantics. *Environmental Modelling and Software, 19*, 219–234.

Argent, R.M., & Rizzoli, A.E. (2004). Development of multi-framework model components. In C. Pahl-Wostl, S. Schmidt, A.E. Rizzoli, & A.J. Jakeman (Eds.), *Transactions of the 2nd Biennial Meeting of the International Environmental Modelling & Software Society* (Vol. 1, pp. 365–370). Osnabrück, Germany: InternationalEnvironmental Modelling and Software Society (iEMSs).

Athanasiadis, I.N., Rizzoli, A.E., Donatelli, M., & Carlini, L. (2006). *Enriching software model interfaces using ontology-based tools*. iEMSs Congress, Vermont, July 2006, http://www.iemss.org/iemss2006/papers/s5/284_Athanasiadis_1.pdf.

Aylor, D. (1982). Modeling spore dispersal in a barley crop. *Agricultural Meteorology, 26*(3), 215–219.

Baier, T. (2007). *The rcom package*. Available online at http://cran.r-project.org/doc/packages/rcom.pdf version 1.5-2.2. Retrieved September 17, 2007.

Baker, C. J., Berry, P. M., Spink, J. H., Sylvester-Bradley, R., Griffin, J. M., Scott, R. K., et al. (1998). A method for the assessment of the risk of wheat lodging. *Journal of Theoretical Biology, 194*, 587–603.

Balderacchi, M., Boccelli, R., & Trevisan, M. (2007). *Tools to assess pesticide environmental fate – Agrochemicals/APES, EPRIP 2 and FitoMarche software*. Pavia, Italy: La Goliardica Pavese, pp. 142, ISBN 978-88-7830-477-2.

Barrett, P. D., Laidlaw, A. S., & Mayne, C. S. (2005). GrazeGro, a European herbage growth model to predict pasture production in perennial ryegrass swards for decision support. *European Journal of Agronomy, 23*, 37–56.

Bastiaans, L. (1991). Ratio between virtual and visual lesion size as a measure to describe reduction in leaf photosynthesis of rice due to leaf blast. *Phytopathology, 81*(6), 611–615.

Bishop, J. (2008). *C# 3.0 design patterns* (p. 314). Sebastopol, CA: O'Reilly.

Blaise, P. H., & Gessler, C. (1992). An extended progeny/parent ratio model. I. Theoretical development. *Journal of Phytopathology, 134*, 39–52.

Braudeau, E. (2006). *Le modèle Kamel*. DDN.FR.001.390019.000.S.P.2006.000.31500. Paris: Agence pour la Protection des Programmes.

Braudeau, E., & Mohtar, R. H. (2006). Modeling the swelling curve for packed soil aggregates using the pedostructure concept. *Journal of the Soil Science Society of America, 70*, 494–502.

Braudeau, E., & Mohtar, R.H. (2009). Modeling the soil system: Bridging the gap between pedology and soil-water physics. *Global and Planetary Change 67*, 51–61.

Brisson, N., Gary, C., Justes, E., Roche, R., Mary, B., Ripoche, D., et al. (2003). An overview of the crop model STICS. *European Journal of Agronomy, 18*(3–4), 309–332.

Brisson, N., Mary, B., Ripoche, D., Jeuffroy, M. H., Ruget, F., Nicoullaud, B., et al. (1988). STICS: A generic model for the simulation of crops and their water and nitrogen balances. I. Theory and parameterization applied to wheat and corn. *Agronomie, 18*, 311–346.

Calvière, I., & Duru, M. (1999). The effect of N and P fertilizer application and botanical composition on the leaf/stem ratio patterns in spring in Pyrenean meadows. *Grass and Forage Science, 54*, 255–266.

Campbell, G. S. (1985). *Soil physics with BASIC* (p. 150). Amsterdam: Elsevier.

Carlini, L., Bellocchi, G., & Donatelli, M. (2006). Rain, a software component to generate synthetic precipitation data. *Agronomy Journal, 98*, 1312–1317.

Carsel, R.F., Imhoff, J.C., Kummel, P.R., Cheplick, J.M., & Donigan, A.S.J. (1988). *PRZM-3 A model for predicting pesticide and nitrogen fate in the crop root and unsatured soil zones: User manual for release 3.0*. Athens, GA: National Exposure Research Laboratory, Office of Research and Development, U.S. Environmental Protection Agency.

Castelan-Estrada, M. (2001). *Growth and dry matter allocation in grapevine (Vitis vinifera): Radiation use efficiency and energetic costs*. Ph.D. thesis, INA Paris-Grignon, France, 121 p.

Confalonieri, R., Acutis, M., Bellocchi, G., Cerrani, L., Tarantola, S., Donatelli, M., et al. (2006). Exploratory sensitivity analysis of CropSyst, WARM and WOFOST: A case-study with rice biomass simulations. *Italian Journal of Agrometeorology, 3*, 17–25.

Confalonieri, R., Bellocchib, G., & Donatelli, M. (2010). A software component to compute agro-meteorological indicators. *Environmental Modelling and Software*. (in press) Available from http://dx.doi.org/10.1016/j.envsoft.2008.11.007

Cooley, K.R. (1980). Erosivity "R" for individual design storms. In W.G. Knisel (Ed.), *CREAMS: A field-scale model for chemicals, runoff, and erosion from agricultural management systems* (pp. 386–397) (USDA-SEA Conservation Research Rep. No. 26). Washington, DC: USDA.

Corbeels, M., McMurtrie, R. E., Pepper, P. A., & O'Connell, A. M. (2005). A process-based model of nitrogen cycling in forest plantations. Part I. Structure, calibration and analysis of the decomposition model. *Ecological Modelling, 187*(4), 426–448.

Cwalina, K., & Abrams, B. (2006). Aggregate components. In *Framework design guidelines: Conventions, idioms, and patterns for reusable .NET libraries* (pp. 235–271). Westford, MA: Addison-Wesley.

David, O., Markstrom, S. L., Rojas, K. W., Ahuja, L. R., & Schneider, W. (2002). The object modelling system. In L. R. Ahuja, L. Ma & T. A. Howell (Eds.), *Agricultural system models in field research and technology transfer* (pp. 317–344). Boca Raton, FL: Lewis.

Del Furia, L., Rizzoli, A., & Arditi, R. (1995). Lakemaker: A general object-oriented software tool for modelling the eutrophication process in lakes. *Environmental Software, 10*(1), 43–64.

Di Guardo, A., Donatelli, M., & Botta, M. (2007). Two framework components to simulate biophysical systems. In *Proceedings of Farming Systems Design 2007, Catania, Italy, 10–12 September 2007*.

Donatelli M., Acutis M., Bregaglio S., Rosenmund A., & Casellas E. (2009). A framework and a software component to simulate agricultural management. Environmental Modelling and Software (submitted).

Donatelli, M., Bellocchi, G., & Carlini, L. (2006a). A software component for estimating solar radiation. *Environmental Modelling and Software, 21*(3), 411–416.

Donatelli, M., Bellocchi, G., & Carlini, L. (2006b). Sharing knowledge via software components: Models on reference evapotranspiration. *European Journal of Agronomy, 24*(2), 186–192.

Donatelli, M., Bolte, J., van Evert, F., & Wang, W. (2003). Which software designs for evolution. In M.K. van Ittersum & M. Donatelli (Eds.), Modelling cropping systems: Science, software and applications. *European Journal of Agronomy, 18*, 193–195.

Donatelli M., Bellocchi G., Habyarimana E., Confalonieri R., & Micale F. (2009). An extensible model library for generating wind speed data. Computers and Electronics in Agriculture, *69* (2009) 165–170.

Donatelli, M., Bellocchi G., Habyarimana E., Bregaglio S., Confalonieri R. & Baruth B., (2009b). CLIMA: a weather generator framewor. In Anderssen, R.S., R.D. Braddock and L.T.H. Newham (eds) 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation, July 2009, pp. 2377–2383. ISBN: 978-0-9758400-7-8. http://www.mssanz.org.au/modsim09/C3/donatelli_C3a.pdf

Donatelli, M., Confalonieri R., Cerrani I., Fanchini D., Acutis M., Tarantola S. & Baruth B., (2009c). LUISA (Library User Interface for Sensitivity Analysis): a generic software component for sensitivity analysis of bio-physical models. In Anderssen, R.S., R.D. Braddock and L.T.H. Newham (eds) 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation, July 2009, pp. 2377–2383. ISBN: 978-0-9758400-7-8.
http://www.mssanz.org.au/modsim09/C3/donatelli_C3b.pdf

Donatelli, M., Omicini, A., Fila, G., & Monti, C. (2004). Targeting reusability and replaceability of simulation models for agricultural systems. In S.E. Jacobsen, C.R. Jensen, & J.R. Porter (Eds.), *Proceedings of the 8th European Society for Agronomy Congress* (pp. 237–238), 11–15 July, Copenhagen, Denmark.

Donatelli, M., & Rizzoli, A. (2008). A design for framework-independent model components of biophysical systems International Congress on Environmental Modelling and Software iEMSs 2008. *Proceedings of the iEMSs Fourth Biennial Meeting, Barcelona, Catalonia*, 7–10 July 2008, pp. 727–734.

Duru, M. (2008). Improvement of time-driven models of lamina cocksfoot digestibility by a process-based model to take account of plant N nutrition and defoliation. *Journal of Agronomy and Crop Science, 194*(5), 401–412.

Duru, M., Adam, M., Cruz, P., Martin, G., Ansquer, P., Ducourtieux, C., Jouany, C., Theau, J.P., & Viegas, J. (2009a). Modelling above-ground herbage mass for a wide range of grassland community types. *Ecological Modelling, 220*, 209–225.

Duru, M., Al Haj Khaled, R., Ducourtieux, C., Theau, J.P., Quadros, F., & Cruz, C. (2009). Do plant functional types based on leaf dry matter content allow characterizing native grass species and grasslands for herbage growth pattern? *Plant Ecology, 201*, 421–433.

Duru, M., Cruz, P., Haj Khaled, R., Ducourtieux, C., & Theau, J. P. (2008). Relevance of plant functional types based on leaf dry matter content for assessing digestibility of native grass species and species-rich grassland communities in spring. *Agronomy Journal, 100*, 1622–1630.

Ferrer-Alegre, F., & Stockle, C. O. (1999). A model for assessing crop response to salinity. *Irrigation Science, 19*, 15–23.

Georgiadis, T., Rossi, S., & Nerozzi, F. (1995). Inferring ozone deposition on agricultural surfaces: An application to herbaceous and tree canopies. *Water, Air, and Soil Pollution, 84*, 117–128.

Green, W.H., & Ampt, G.A. (1914). Studies on soil physics. *Journal of Agricultural Science, 4*(1), 1–24. Hydrology. *Transactions American Society Agricultural Engineering, 20,* 1100–1104.

Hearn, A.B. (1994). The principles of cotton water relations and their application in management. In G.A. Constable & N.W. Forrester (Eds.), *Proceedings World Cotton Research Conference, 1st, Brisbane, Australia* (pp. 66–92), 14–17 February 1994. Melbourne, Australia: CSIRO.

Hillyer, C., Bolte, J., van Evert, F., & Lamaker, A. (2003). The MODCOM modular simulation system. *European Journal of Agronomy, 18*(3–4), 333–343.

Jantunen, A. P. K., Trevisan, M., & Capri, E. (2005). Computer models for characterizing the fate of chemicals in soil: Pesticide leaching models and their practical applications. In J. Alvarez-Bendì & R. Muñoz-Carpena (Eds.), *Soil-Water-Solute process characterisation: An integrate approach* (pp. 715–756). Boca Raton, FL: CRC Press.

Johnsson, H., Bergström, L., Jansson, P. E., & Paustian, K. (1987). Simulated nitrogen dynamics and losses in a layered agricultural soil. *Agriculture, Ecosystems, and Environment, 18*, 333–356.

Jones, J. W., Hoogenboom, G., Porter, C. H., Boote, K. J., Batchelor, W. D., Hunt, L. A., et al. (2003). The DSSAT cropping system model. *European Journal of Agronomy, 18*(3–4), 235–265.

Jones, J. W., Keating, B. A., & Porter, C. H. (2001). Approaches to modular model development. *Agricultural Systems, 70*, 421–443.

Karlberg, L., Ben-Gal, A., Jansson, P.-E., & Shani, U. (2006). Modelling transpiration and growth in salinity-stressed tomato under different climatic conditions. *Ecological Modelling, 190*, 15–40.

Keating, B. A., Carberry, P. S., Hammer, G. L., Probert, M. E., Robertson, M. J., Holzworth, D., et al. (2003). An overview of APSIM, a model designed for farming systems simulation. *European Journal of Agronomy, 18*(3–4), 267–288.

Lavorel, S., & Garnier, E. (2002). Predicting changes in community composition and ecosystem functioning from plant traits, revisiting the Holy Grail. *Functional Ecology, 16*, 545–556.

Magarey, R. D., Sutton, T. B., & Thayer, C. L. (2005). A simple generic infection model for foliar fungal plant pathogens. *Phytopathology, 95*(1), 92–100.

Makowski, D., Hillier, J., Wallach, D., Andrieu, B., & Jeuffroy, M. H. (2006). Parameter estimation for crop models. In D. Wallach, D. Makowski & J. W. Jones (Eds.), *Working with dynamic crop models* (pp. 101–150). Amsterdam: Elsevier.

Martin, P., Mohtar, R. H., Clouvel, P., & Braudeau, E. (2006, July). *Modeling soil-water dynamics for diverse environmental needs*. Vermont: iEMSs Congress.

McCall, D. G., & Bishop-Hurley, G. J. (2003). A pasture growth model for use in a whole-farm dairy production model. *Agricultural Systems, 76*, 1183–1205.

Mesketer, S. J. (2004). *Design patterns in C#*. Boston: Addison-Wesley.

Meyer, B. (1991). Design by contract. In D. Mandrioli & B. Meyer (Eds.), *Advances in object-oriented software engineering*. Englewnod Cliffs, NJ: Prentice Hall.

Monteith, J. L. (1977). Climate and the efficiency of crop production in Britain. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 281*, 277–294.

Monteith, J. L., & Unsworth, M. H. (1990). *Principles of environmental physics*. Woburn, MA: Butterworth-Heinemann.

Morgan, R. P. C., Quinton, J. N., Smith, R. E., Govers, G., Poesen, J. W. A., Auerswald, K., et al. (1998). The European soil erosion model (EUROSEM): A dynamic approach for predicting sediment transport from fields and small catchments. *Earth Surface Processes and Landforms, 23*, 527–544.

Mulia, R. (2005). *Modélisation tri-dimensionnelle de la croissance du système racinaire des plantes en milieu hétérogène avec l'approche de l'automate voxellaire*. Ph.D. thesis, USTL, Montpellier 2, p. 86.

Mulia, R., & Dupraz, C. (2006). Unusual fine root distributions of two deciduous tree species in southern France: What consequences for modelling of tree root dynamics*? Plant and Soil, 281*(1/2), 71–85.

Neitsch, S.L., Arnold, J.G., Kiniry, J.R., Williams, J.R., & King, K.W. (2002). *Soil and water assessment tool*. Theoretical Documentation. Temple, TX: Grassland, Soil and Water Research Laboratory, p. 506.

Nendel, C., & Kersebaum, K. C. (2004). A simple model approach to simulate nitrogen dynamics in vineyard soils. *Ecological Modelling, 177*, 1–15.

Ollat, N., Diakou-Verdin, P., Garde, J. P., Barrieu, F., Gaudillière, J. P., & Moing, A. (2002). Grape berry development: A review. *Journal International des Sciences de la Vigne et du Vin, 36*(3), 109–131.

Parton, W. J. (2004). Predicting soil temperatures in a shortgrass steppe. *Soil Science, 138*, 93–101.

Pronk, A., Goudriaan, J., Stilma, E., & Challa, H. (2003). A simple method to estimate radiation interception by nursery stock conifers: A case study of eastern white cedar. *Netherlands Journal of Agricultural Science, 51*, 279–295.

R Development Core Team. (2007). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing.

Ritchie, J. T. (1972). Model for predicting evaporation from a row crop with incomplete cover. *Water Resources Research, 85*, 1204–1211.

Ritchie, J.T. (1991). Wheat phasic development. In J. Hanks & J.T. Ritchie (Eds.), *Modelling plant and soil systems.* Agronomy Monographs 31 (pp. 31–54). Madison, WI: ASA, CSSSA, SSSA.

Ritchie, J.T., & Otter, S. (1985). Description and performance of CERES-Wheat: A user oriented wheat yield model. In *ARS Wheat Yield Project. ARS-28* (pp. 159–175). Springfield, VA: National Technology Information Service.

Rizzoli, A. E., Donatelli, M., Athanasiadis, I., Villa, F., Muetzelfeldt, R., & Huber, D. (2005). Semantic links in integrated modelling frameworks. *Mathematics and Computers in Simulation, 78*, 412–423.

Rizzoli, A.E., Donatelli, M., Muetzelfeldt, R., Otjens, T., Svennson, M.G.E., van Evert, F., Villa, F., & Bolte, J. (2004). SEAMFRAME, a proposal for an integrated modelling framework for agricultural systems. In S.E. Jacobsen, C.R. Jensen, & J.R. Porter (Eds.), *Proceedings of the 8th European Society for Agronomy Congress* (pp. 331–332), 11–15 July, Copenhagen, Denmark.

Saltelli, A.,Tarantola, S., Campolongo, F. & Ratto, M. (2004). Sensitivity analysis in practice: a guide to assessing scientific models. Chichester, England: John Wiley & Sons Ltd.

Saxton, K. E., & Rawls, W. J. (2006). Soil water characteristic estimates by texture and organic matter for hydrologic solutions. *Journal of the Soil Science Society of America, 70*, 1569–1578.

Schapendonk, A. H. C. M., Stol, W., Van Kraalingen, D. W. G., & Bouman, B. A. M. (1998). LINGRA, a sink/source model to simulate grassland productivity in Europe. *European Journal of Agronomy, 9*, 87–100.

M.E. Shibu, P.A. Leffelaar, H. van Keulen, P.K. Aggarwal (2009). LINTUL3, a simulation model for nitrogen-limited situations – application to rice European Journal of Agronomy (submitted).

Shimono, H., Hasegawa, T., Moriyama, M., Fujimura, S., & Nagata, T. (2005). Modeling spikelet sterility induced by low temperature in rice. *Agronomy Journal, 97*, 1524–1536.

Sitch, S., Cox, P. M., Collins, W. J., & Huntigford, C. (2007). Indirect radiative forcing of climate change through ozone effects on the land-carbon sink. *Nature, 448*, 791–795.

Smith, R. E., & Parlange, J. Y. (1978). A parameter-efficient hydrologic infiltration model. *Water Resources Research, 14*, 533–538.

Soil Conservation Service. (1972). Section 4: Hydrology. In *National Engineering Handbook*. SCS.

Spiker, E. C., Hosker, R. P., Comer, V. J., White, J. R., Werre, R. W., Jr., Harmon, F. L., et al. (1992). Environmental chamber for study of the deposition flux of gaseous pollutants to material surfaces. *Atmospheric Environment, 26*, 2885–2892.

Stockle, C. O., Donatelli, M., & Nelson, R. (2003). CropSyst, a cropping systems simulation model. *European Journal of Agronomy, 18*(3–4), 289–307.

Streck, N. A., Weiss, A., Xue, Q., & Baenziger, P. S. (2003). Improving predictions of developmental stages in winter wheat: A modified Wang and Engel model. *Agricultural and Forest Meteorology, 115*, 139–150.

Szypersky, C., Gruntz, D., & Murer, S. (2002). *Component software - beyond object-oriented programming* (2nd ed.). London: Addison-Wesley.

Tiktak, A., Van den Berg, F., Boesten, J.J.T.I., Van Kraalingen, D., Leistra, M., & Van der Linden, A.M.A. (2001). *Manual of FOCUS PEARL v 1.1.1. RIVM Report 711401008*, Alterra Report 28 (p. 144). Bilthoven, The Netherlands: RIVM.

Trevisan, M., Sorce, A., Balderacchi, M., & Di Guardo, A. (2007). A software component to simulate agro-chemicals fate. In *Proceedings of Farming Systems Design 2007*, Catania, Italy, 10–12 September 2007.

Van Dam, J.C., Huygen, J., Wesseling, J.G., Feddes, R.A., Kabat, P., Van Walsum, P.E.V., Groenendijk, P., & Van Diepen, C.A. (1997). *Theory of SWAP version 2.0. Report 71*. Wageningen, The Netherlands: Department of Water Resources, WAU.

Van Evert, F., & Lamaker, A. (2007). The MODCOM framework for component-based simulation. In *proceedings of Farming Systems Design 2007*, Catania, Italy, 10–12 September, 2007.

Van Ittersum, M. K., Ewert, F., Heckelei, T., Wery, J., Alkan Olsson, J., Andersen, E., et al. (2008). Integrated assessment of agricultural systems – A component-based framework for the European Union (SEAMLESS). *Agricultural Systems, 96*(1–3), 150–165.

Van Ittersum, M. K., Leffelaar, P. A., Van Keulen, H., Kropff, M. J., Bastiaans, L., & Goudriaan, J. (2003). On approaches and applications of the Wageningen crop models. *European Journal of Agronomy, 18*(3–4), 187–393.

Van Keulen, H., & Seligman, N.G. (1987). Simulation of water use, nitrogen nutrition and growth of a spring wheat crop. *Simulation Monographs*. Wageningen, The Netherlands: Pudoc.

Van Keulen, H., & Wolf, J. (1986). Modelling of agricultural production: Weather soils and crops. *Simulation Monographs*. Wageningen, The Netherlands: Pudoc.

Villa F., Donatelli, M., Rizzoli, A., Krause, P., Kralisch, S., & Van Evert, F.K. (2006, July). Declarative modelling for architecture independence and data/model integration: A case study. iEMSs congress, Vermont.

Vivin, Ph, Castelan, M., & Gaudillère, J. P. (2002). A source/sink model to simulate seasonal allocation of carbon in grapevine. *Acta Horticulturae, 584*, 43–56.

Von Hoyningen-Huene, J. (1981). *Die Interzeption des Niederschlags in landwirtschaftlichen Pjanzenbestanden*. Arbeitsbericht Deutscher Verband fur Wasserwirtschaft und Kulturbau, DVWK, Braunschweig.

Wadia, K. D. R., & Butler, D. R. (1994). Relationship between temperature and latent periods of rust and leaf-spot diseases of groundnut. *Plant Pathology, 43*, 121–129.

Waggoner, P. E. (1973). The removal of Helminthosporium maydis spores by wind. *Phytopathology, 63*(10), 1252–1255.

Waggoner, P. E., & Horsfall, J. G. (1969). EPIDEM. A simulator of plant disease written for a computer. *Bulletin of the Connecticut Agricultural Experiment Station, 698*, 80.

Wermelinger, B., & Koblet, W. (1990). Seasonal growth and nitrogen distribution in grapevine leaves, shoots and grape. *Vitis, 29*, 15–26.

Williams, J.R., & Berndt, H.D. (1977). Sediment Yield Prediction Based on Watershed Hydrology. *Transactions of the American Society of Agricultural Engineers*, 20, 1100–1104.

Williams, J. R., Jones, C. A., Kiniry, J. R., & Spanel, D. A. (1989). The EPIC crop growth model. *Transactions of the American Society of Agricultural Engineering, 32*, 497–511.

Woolhiser, D.A., Smith, R.E., & Goodrich, D.C. (1990). KINEROS, a kinematic runoff and erosion model: Documentation and user manual. United States Department of Agriculture, ARS-77.

Wosten, J. H. M., Lilly, A., Nemes, A., & Le Bas, C. (1999). Development and use of a database of hydraulic properties of European soils. *Geoderma, 90*, 169–185.

Zadoks, J.C., & Schein, R.D. (1979). *Epidemiology and plant disease management* (p. 427). London: Oxford University Press.