RESEARCH ARTICLE

FSPM-P: towards a general functional-structural plant model for robust and comprehensive model development

Michael HENKE (✉)[1], Winfried KURTH[1], Gerhard H. BUCK-SORLIN[2]

1   Department of Ecoinformatics, Biometrics and Forest Growth, University of Göttingen, Göttingen 37077, Germany
2   UMR1345 Institut de Recherche en Horticulture et Semences (IRHS), AGROCAMPUS OUEST Centre d'Angers, Angers 49045, France

Abstract   In the last decade, functional-structural plant modelling (FSPM) has become a more widely accepted paradigm in crop and tree production, as 3-D models for the most important crops have been proposed. Given the wider portfolio of available models, it is now appropriate to enter the next level in FSPM development, by introducing more efficient methods for model development. This includes the consideration of model reuse (by modularisation), combination and comparison, and the enhancement of existing models. To facilitate this process, standards for design and communication need to be defined and established. We present a first step towards an efficient and general, i.e., not species-specific FSPM, presently restricted to annual or bi-annual plants, but with the potential for extension and further generalization.

Model structure is hierarchical and object-oriented, with plant organs being the base-level objects and plant individual and canopy the higher-level objects. Modules for the majority of physiological processes are incorporated, more than in other platforms that have a similar aim (e.g. photosynthesis, organ formation and growth). Simulation runs with several general parameter sets adopted from the literature show that the present prototype was able to reproduce a plausible output range for different crops (rapeseed, barley, etc.) in terms of both the dynamics and final values (at harvest time) of model state variables such as assimilate production, organ biomass, leaf area and architecture.

1   Introduction

Current crop growth models are often based on a selection of general processes describing the mechanisms of primary production. Generally, in these models factors determining potential, attainable and actual crop growth are distinguished, allowing the same model to be used for a variety of crop species, given the availability of a standard set of crop parameters [1].

In contrast to these process-based models, functional-structural plant modelling (FSPM) has its origin in purely structural modelling, and within this paradigm models are developed in a much more *ad hoc* way. Developers of such models are often plant biologists who are keen to explore the impact of plant architecture (organ geometry and topology) on a limited range of physiological effects, e.g., the effect of leaf angle distribution on canopy radiation interception. These workers are often lacking experience in programming yet have a clear overview of the structure and scope of their model. Another group consisted of programmers and computer scientists who are interested in biological systems considers it as a challenge for the application of the rule-based paradigm. Thus, while plant biologists use an FSPM approach to study the effect of a static architecture on light interception and leaf photosynthesis, computer scientists study the way complex tree architectures could be created using a

very limited set of production rules. Most physiological functions that are currently used in crop models could be used in the same general way in FSPM, and structures, such as plant organs, could be defined generally and then implemented for a crop species.

Current FSPMs of crop plants (e.g., for peach [2, 3]; rice [4]; cut-rose [5]; rape [6]; barley [7–9]) contain common components and recurring parts (e.g., for photosynthesis, growth and extension of organs, build-up of the structure through formation of phytomers at the shoot tip and through branching), which could be generalized and re-used as subsystems. One possible solution to benefit from former models is a prototype as a base for new models.

FSPMs with a generic character are not numerous. Amap-Sim [10] is in its core a purely structural model, allowing the linking of functional components as external programme modules. Based on the notion of physiological age, it was primarily adapted to trees. However, it does not inherently support the feedback of carbon assimilation on growth and structural development, which makes it less useful for crop plant simulations. GreenLab [11] uses the concept of physiological age in its structural part; it was used to model several crop plant species. Furthermore, the feedback of assimilates on structural growth was included in the advanced version GL3 [12]. Because of the simplified description of source functions, it was considered as "intermediate between FSPM and (purely) process-based models" [12]. Breckling [13] designed an FSPM for a generic, modular plant and implemented it in the object-oriented language Simula. However, to adapt it to real crop, the Simula source code has to be modified. Finally, LIGNUM [14] uses annual time steps for growth and was designed for Scots pine in its first version; later it was adapted to other tree species. These adaptations require changes in the code again.

Here we present an FSPM prototype which goes a step further than the FSPM approaches described in the previous paragraph: while plant architecture is still largely descriptive (i.e., organ geometry and arrangement is input to the model), the majority of processes related to the functioning of sources and sinks are implemented in a generic way, allowing the computation of resource allocation according to the demand of each organ. The model is written in the rule-based language XL and implemented on the software platform GroIMP (see Section 2.1). This model uses an object library in which each botanical object is provided with predefined state variables and methods representing internal processes (photosynthesis, growth, maintenance and growth res-

piration, storage and remobilisation of assimilates). Because of its structured, object-oriented design, modular set-up and a user manual provided with it, it is easy to parametrize, use and extend. The prototype has not yet reached its final degree of generality; some default values and procedures are chosen arbitrarily in order to allow the user to get started rapidly and will be replaced by more general or exchangeable parts in a future version. In its current version, our model and this paper are meant to provide scientists and students of the plant sciences with an easy access to the FSPM paradigm, which might be a valuable additional tool for hypothesis testing.

2 Materials and methods

2.1 Modelling language and platform

The present FSPM-P (FSMP-Prototype) is written using the modelling language XL (eXtended L-System modelling language) [15], a rule-based language which supports the specification of graph grammars generalizing L-systems [16, 17] and which is at the same time a superset of the language Java. Hence each Java programme can easily be embedded in an XL programme. The modelling platform GroIMP is platform-independent, open-source and freely available[1]. GroIMP is employed for model implementation and visualisation. It is designed as an integrated platform which incorporates modelling, simulation, visualisation and user interaction, and provides a compiler and development tools for XL [15].

2.2 General features of the FSPM-Prototype project

The FSPM-Prototype project comprises two elements, the FSPM-Prototype model (FSPM-P) (current version: 0.4) and a user manual as free download from the model gallery at www.grogra.de. The model is subdivided into separate modules: a main file for model initiation and control; a file for defining objects (such as plant organs) and their properties; a library of photosynthesis rate models to be coupled with leaf objects; global parameter definition; a file containing auxiliary tools and functions like charts. To make FSPM-P an accessible and comprehensible tool, an extensive user manual which provides a detailed *Model description* was written.

The FSPM-P is a fairly extensive set of XL modules and Java implementations comprising the description of a fairly comprehensive set of biophysical and physiological processes such as radiation interception, photosynthesis, growth

---

[1] www.grogra.de

and development. The hierarchical scale at which the model is implemented is the same as that of the organ, but processes can also be aggregated at the plant individual scale.

In the following sections, we will describe some features of the current model: definitions for plant organs, work flow, growth and development, the latter being based on source (local photosynthesis of assimilates, storage of assimilates locally and in a central pool) and sink functions (reallocation of assimilates for growth as a function of sink strength, i.e., relative potential growth rate with the source/sink ratio used to steer growth and branching).

2.3　Plant definition

Within the FSPM-P model a plant species is defined by three files, 1) a parameter file, with species-specific parameters mainly for growth and photosynthesis, 2) all rules for morphology, cutting, transport, and organ update etc., are collected in a rule file, 3) and a module file listing predefined plant organs. In addition, there are different hierarchical scales within the plant organ definition: basic organs (seed, root, meristem, bud, leaf, internode, flower, fruit, etc.) and organ aggregations (individual and shoot). They contain, e.g., standard variables and summary functions based on XL-queries to get fast information about the internal plant state.

The object-oriented design of the FSPM-P with its strict separation of species-, parameter-, and infrastructure-specific parts, allows to simulate more than one species at once, which can be done by adding an additional file set for the new species and to activate its simulation in the main model loop. Besides, further things like arrangement of the individual plants and their interactions needs to be defined by the end user. Currently, shading effects between different species are the only emergent process that the FSPM-P provides. Other processes like sensing, independent of the above or below ground, concurrence about nutrients or any stress effects need to be implemented by the user.

Plant structure and topology are based on measurements. They are defined by morphological rules and therefore input to the model. For the following description of the prototype, hypothetical "observations" based on real data are used.

2.4　Model work flow

During initialisation global parameters and variables are loaded, direct and diffuse light sources (sun and sky light) as well as a single plant or plant stand are put into the scene with their initial parameters. In a main loop (method `run()`), a single growth step is repeated until the user stops it manu-

ally or after a predefined time. For each growth step, four sub-steps are carried out: update the sun and sky module; run light model; apply rules; update output. Finally, some statistical outputs, e.g., amount of harvested biomass, are generated and pasted into a chart. The work flow in the model is summarized in Fig. 1.
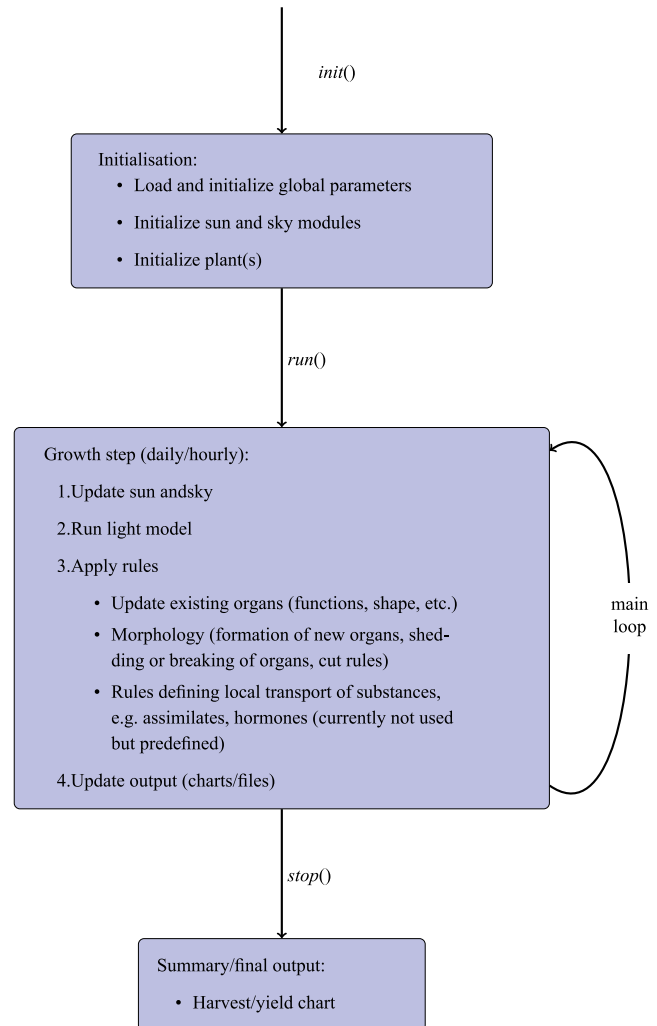
Fig. 1　General model work flow: after the initialisation, the model will be executed during the main loop before final output is generated

The `applyRules()` function is the only species-specific function within the main loop. For each species a user wishes to simulate one such function call needs to be included. Consequently, all simulated species are sharing the same scenario and environment condition, while the type and number of processes as well as their temporal resolution does not need to be the same.

To improve legibility of the code, the `applyRules()` function is also clearly subdivided.

The different methods invoked are described in the following section.

```
protected void applyRules() {
    morphologyRules();
    cutRules();
    transportRules();
    organUpdates();
    otherRules();
}
```

2.5   General processes

According to their different functionalities, there are rules for morphology (formation of new phytomers at the tip of an axis, and branching), cutting/abscission of organs, transport, organ updates (of internal parameters, e.g., length, diameter, mass, as well as processes, e.g., growth and maintenance respiration), and other rules (mainly for information about the current state of the model).

The function `morphologyRules()` comprises the following rules:

1) Germination   If conditions for germination are satisfied, replace seed with root and a meristem (containing the shoot apical meristem). The meristem has three parameters: the plant individual that it belongs to, the rank (running number of phytomers in the shoot, counted from the base), and the branching order. The two last parameters are initialized with 1.

2) Development   The corresponding rule finds all meristem objects that fulfil certain conditions, and replaces them with a phytomer, i.e., an internode, a leaf, and a new meristem or bud. The final rules are analogous to the first bud rule, but replace the bud with a flower, and the flower with a fruit, respectively, if the conditions for these processes are met.

The conditions for bud break are 1) topological: rank and order; 2) light: a bud must absorb more light than a threshold; 3) temperature: mean air temperature must be in a suitable range; 4) the average source/sink ratio of the plant has to be bigger than a user-defined threshold. The latter condition ensures that the plant currently has sufficient reserves to form new phytomers; 5) a bud break probability model, e.g., by a semi-Markov chain; and 6) phyllochron. Finally, as an exceptional case for formation of new sinks in a situation of overproduction of assimilates, sleeping (dormant) buds can be reactivated when a specific average source/sink ratio is reached.

A newly-formed meristem is initialized with a species-specific phyllochron (measured in thermal time units), which expresses the developmental phase between bud initiation and bud break to form a new phytomer; this internal variable is decreased at each organ update by the actual average temperature. When the phyllochron is counted down to zero (or has a negative value), one condition for phytomer production is fulfilled and the rule may be executed.

Growth and development are based on source (leaf photosynthesis of assimilates and release from a storage pool) and sink functions (reallocation of assimilates for growth as a function of relative sink strength, storage in the pool).

Photosynthesis in the model is restricted to leaf blades; photosynthesis of other green organs such as sheaths, stems and walls of immature fruits is currently not considered (however, this would be possible without problems as all these organs implement the organ superclass).

Simplified transport of water is implemented to illustrate the usage of the ordinary differential equation (ODE) framework of GroIMP [18]. An inexhaustible water reservoir provides the water that can be absorbed by the root. The absorbed water is piped through internodes and leaves driven by a temperature sensitive transpiration function within each leaf.

2.6   Radiation model and light interception

GroIMP provides two ways for calculation of light interception: 1) a central processing unit (CPU) based implementation [15, 19] and 2) an implementation able to use multiple devices in parallel inclusive of the graphics processing unit (GPU) called GPUFlux [20]. The user has to choose the method that is used to simulate light distribution and local light interception. These methods are based on a reversed path tracer algorithm with Monte-Carlo integration [21] and use light sources and geometric objects placed into a scene. The selected radiation model is invoked once per simulation step, and is applied to a scene created within the modelling environment GroIMP. GroIMP provides several types of light sources. As default setting, we use a directional light source to simulate direct sun light whereas diffuse sky light is simulated using an array of 72 directional lights positioned regularly in a hemisphere in six circles with twelve lights each, with emitted power densities being a fixed function of the elevation angle [22, 23]. As alternative sky model, an implementation based on Preetham [24] is integrated into GroIMP too. It is planned to provide several established sky models as alternative choices in a future version of FSPM-P. Both the sun and the sky object are dynamically updated at each step as function of the Julian day of the year and the time of the day [h]. The light model is run with two parameters: total number of rays produced by all light sources in the scene, and

the number of times a reflected or transmitted ray is traced. In the default configuration, we recommend to use at least ten million rays for the CPU ray tracer in the daily, a twenty-fourth of it in the hourly run mode and a recursion depth of ten. For the much faster GPU ray tracer, the number of rays can be easily increased up to 200 millions and even more, in order to enhance accuracy of the obtained light distribution.

Once a leaf is formed, it is identified with a label, and its absorbed radiation is determined as a spectrum at a run of the light model. This spectrum is converted from [$W/m2$] to Photosynthetic Photon Flux Density PPFD [$\mu mol PPFD/(m2s)$] by multiplication with a conversion factor (2.275 in the case of daylight [1]).

To simulate the distribution of direct PAR during the day, the position of the sun is computed according to Goudriaan and van Laar [1], and the normal vector representing that position is transformed into a vector representing the orientation of the directional light source, updated at an hourly rate.

The advanced GPUFlux ray tracer [20] supports multiple devices for simultaneous calculations, e.g., all threads of a CPU and, in addition, a GPU, which reduces the time for light calculation dramatically. Besides this significant acceleration, the GPUFlux ray tracer provides the possibility to calculate the full spectrum of light, which opens new application areas, totally as discussed in Subsection 3.3.

2.7 Source implementation

The main carbon sources for a plant in our model are the leaves (after the carbon stored in the seed has been consumed during germination). Intermediate storage and remobilization of starch is considered only in the root organ, where at each time step a small amount (1%–2.5%) of the produced assimilates is stored. This storage pool is used as source only in the last developmental stage, during fruit development, and during times where environmental conditions are unfavourable for growth. (For convenience, it is located in the root organ, though in reality it might be distributed all over the plant).

Integrated into the model is a library of photosynthesis rate models (differing in complexity from simple light-response curves to biochemical Farquhar-type models), which can be selected with a global parameter (see Section 2.8).

At the level of the individual, all produced assimilates of all leaves, minus a certain fraction local demand LD which is stored in the local pool of the leaf for its own growth, are collected only for calculation purposes in a temporary assimilate pool AP [$g/time$]:

$$AP_t = (1 - LD) \times \sum_{i=1}^{n} PS_i. \tag{1}$$

This is done automatically at each time step by calling the `update()` function in the Individual module, see FSPM-P user manual for more details.

In the current setting, the dynamics of the source is characterized by five phases. In the first phase, initial carbon is provided by the seed, which is rapidly exhausted during germination in the second phase. After unfolding of the first leaves, photosynthesis commences. During the third phase, source and sink are in balance, and the temporary assimilate pool AP is emptied at each step (source/sink ratio fluctuates around one). In the fourth phase of vegetative establishment, source strength is bigger than sink demand and assimilate reserves are stored in the storage pool. During the fifth phase of maturity, fruit formation takes place, and for this the storage pool is used as a further source in addition to the assimilates provided by photosynthesis at each step, but which are declining due to leaf ageing. Feedback inhibition of the photosynthesis rate due to a local excess in assimilates (low sink strength), has not been implemented.

2.8 Photosynthesis models

The temporal resolution of our model can currently be switched by the user between daily and hourly run mode. The required weather file is automatically loaded and used as an input to the photosynthesis model, containing daily or hourly values of mean temperature, global radiation, and relative humidity. If only daily totals of global radiation are available, the expected value for a given hour of the day can be estimated using a sine function [1], assuming atmospheric transmissivity to be a function of global daily radiation and solar elevation, as described by Gijzen [25].

The model's runtime is only restricted by the availability of weather data. Currently, the model provides only a single weather file with daily values from the weather station Haarweg, Wageningen, and the Netherlands[2] (366 days recorded from January 1, 2008 to December 31, 2008), but weather files comprising several years can be used, too.

The method `getPAR()` defined in the leaf module is used to calculate the photosynthetically active radiation PAR [$\mu mol photons/(m2s)$], by taking the actual absorbed radiation and dividing it by the leaf area.

The FSPM-Prototype provides a portfolio of nine photosynthesis rate models, three versions of biochemical leaf photosynthesis rate models considering leaf temperature, PAR,

---

[2] http://www.met.wau.nl/haarwegdata/

CO2 concentration and leaf energy balance: the LEAFC3-N photosynthesis model [26] with consideration of nitrogen [27, 28], Baldocci [29], and the model by Kim and Lieth [30]; furthermore, models based on simple light-response curves [31–37] are included.

The user can select one of these photosynthesis models in the global parameter file, to be used in the model runs.

2.9 Sink activities and their relationship to the source

The timing and growth duration of active sinks drives the conversion of assimilates to harvestable dry matter. In our FSPM approach, the overall control of sink activity is prescribed by growth and development rules, and the overall biomass production is an emergent property of the integration of these rules applied to the growing structure over simulated time, see Fig. 2. In addition, the rate of extension of each organ is described by a sigmoid growth function, e.g., the beta growth function [38]:

$$ w_t = w_{max}(1 + \frac{t_e - t}{t_e - t_m})(\frac{t}{t_m})^{\frac{t_e}{t_e-t_m}}, \tag{2} $$

with $0 \leqslant t_m \leqslant t_e$, where w_{max} is a maximum value of w_t, reached at time t_e, and t_m is the time when growth rate reaches its maximum.
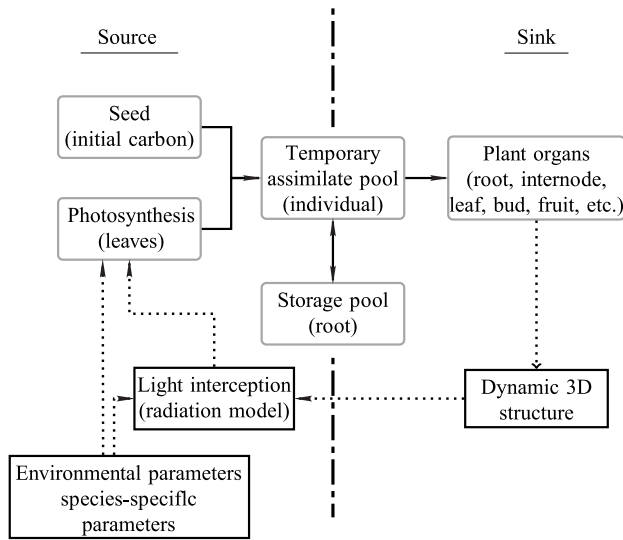
Fig. 2 Schematic overview of source/sink relationship used within the model. After the seed storage is exhausted and the first leaves are developed, photosynthesis takes over as main source process

Here w_{max}, t_m, and t_e are organ-dependent input values, which should be based on real measurements for a given species. Typically, such a growth function also depends on the (acropetal) rank of the leaf or internode (as has been shown for barley by Buck-Sorlin 2002). For FSPM-P, we use hypothetical but realistic values, as shown in Fig. 5(b) for intern-

odes.

The sink strength of a growing organ i at time t can be approximated by its potential growth rate $PGR_{i,t}$, which is the instantaneous increment in dry matter w and can be described by the derivative of the above function:

$$ PGR_{i,t} = \frac{dw_{i,t}}{dt} = c_{max}(\frac{t_e - t}{t_e - t_m})(\frac{t}{t_m})^{\frac{t_m}{t_e-t_m}}, \tag{3} $$

where c_{max} is the maximum growth rate at time t_m [38]. The method `getPGR()` is used to compute the potential growth rate in each organ [$drymass/time$].

As an alternative, other preimplemented growth functions such as Chapman-Richards [39] or a logistic function are provided.

Global sink demand sd [$drymass/time$] is defined as the sum of all potential growth rates PGR of concurrently growing organs:

$$ sd = \sum_{i=1}^{n} PGR_{i,growingorgan}. \tag{4} $$

The relative sink strength RSS [$-$] is calculated for each organ i by:

$$ RSS_i = PGR_i/sd. \tag{5} $$

Multiplication of RSS with the temporary assimilate pool AP [$g/time$] results in the actual/realized growth rate AGR [$g/time$] thereby assuming that AGR cannot be bigger than PGR:

$$ AGR_i = \min(PGR_i, RSS_i * AP_t). \tag{6} $$

In the model, this is implemented for each organ in the `getAGR()` method, where AP is calculated using the method `getTemporaryAssimilatePool()` of the associated individual.

Once growth of an organ takes place, the actual growth AG is added to the $dryWeight$ of each organ, and the temporary assimilate pool is updated accordingly. The unused assimilates at time t is the difference in all assimilates available for growth and sum of respiration losses at the same time step t:

$$ AP_{t+1} = AP_t - \sum_{i=1}^{n} R_{i,t}, \tag{7} $$

where respiration R for an organ i at time t is:

$$ R_{i,t} = MR_{i,t} \times DW_{i,t} + GR_{i,t}. \tag{8} $$

Maintenance respiration MR is computed as an organ-specific fixed proportion of structural biomass, whereas growth respiration GR is defined as the amount of assimilates [g] respired when producing one gram of new biomass [40]. It can be conveniently expressed as a conversion factor, ($g[glucose]/g[newdryweight]$), i.e., the total amount of assimilates per gram new biomass [$-$]. Thus GR is proportional

to the growth rate as described in Goudriaan and van Laar [1]. Both terms are subtracted from the temporary assimilate pool at each step.

If the temporary assimilate pool is not completely exhausted, the excessive assimilates will be added to the storage pool. This storage pool will be activated if the environmental conditions cause an emergency situation for the plant or the fruit formation.

Each plant organ module implements an `update()` function with two parameters: the amount of absorbed radiation and the current mean temperature. At each call of this function, the internal age counter is increased and the carbon budgets are updated as described above.

2.10   Vegetative and generative development

To simulate vegetative and generative development, a small set of growth, developmental and branching rules is repetitively applied to a Bud module and all of its ensuing organs, leading to the visible phenotype. This type of repetitive application of rules is straightforwardly implemented in the rule-based language XL which supports the specification of graph grammars generalizing L-systems [16, 17]. The structural framework created thus is used to simulate and analyse the dynamics of assimilate flow as dictated by local (potential) growth rates and assimilate availability in the temporary assimilate pool. The model simulates phenology, including germination, seedling stage, juvenile (vegetative) and adult (generative) plant, and finally harvest maturity.

Formation of a new organ from a meristem occurs after some intrinsic delay (phyllochron). The main stem and tillers are created within the limits given by topological parameters (i.e., maximum rank and order). A new leaf is formed with an initial dry weight which is converted to the initial length and diameter, plus a new bud initiated at the tip of the shoot, and the rank increased by one. At the same time, the phyllochron is set to its initial value (as specified by a species-specific parameter PHYLLOCHRON).

The potential extension and final dimension of organs (leaves, internodes, etc.) depend upon their rank and age, while the actually achieved dimensions are also a function of sink competition and assimilate availability, as described in Section 2.9. Leaf dimensions are determined using the beta growth function [38], calculating dry matter increment as a function of time, and dry matter is then converted into leaf shape (length and width) using a constant conversion factor for simplicity.

Once the generative stage is attained, flower formation takes place, followed by fruit formation according to a user-defined fertilisation rate which uses simple stochastic mechanisms. Fruits / seeds formed from flowers will grow and change their colour according to the stage of maturity attained, limited by potential growth rate.

2.11   Source/sink ratio for model regulation

A dynamically calculated average source/sink ratio SSR (calculated over a number of previous steps), which exhibits a range of values (usually between 0.1 and 1.1), is used to control the carbon budget in the model [41–44].

The idea is to keep source and sink in balance and to up- or-down regulate the average SSR in such a way that it stays at roughly a value of one. Depending on the value that SSR attains, sink or source regulation in the model takes place in different ways: if the SSR gets too high, the source strength is decreased by decreasing photosynthetic efficiency. Alternatively, sink strength is increased by increasing the number of growing organs (bud break) or their potential growth rate, and by increasing storage of assimilates in the temporary carbon pool.

When source capacity exceeds global demand (i.e., by all growing sinks), a possible measure is the down-regulation of the source, specifically the photosynthetic efficiency, by multiplying the result of the photosynthesis function with a factor. This regulation factor is based on the difference between the average source/sink ratio and one (1-avg (SSR)). Otherwise, if environmental conditions turn very unfavourable and if then as, a consequence, assimilate production is strongly reduced, photosynthetic efficiency cannot be up-regulated again to counterbalance the unfavourable conditions.

Another possibility for a plant to react to a surplus of assimilates is to produce new sinks by increasing the rate of bud break, thus creating new shoots. Conversely, as a reaction to a low source/sink ratio (high sink demand or low source capacity, or both), the photosynthetic efficiency can be increased (see above), and weak sinks can be removed from the plant (e.g., flower or fruit abortion), or PGR of organs reduced.

2.12   Implementation of processes and module communication

All processes are implemented as functions inside each organ definition with an organ-specific parameterization (Table 1).

According to the organ superclass, all organs are having processes implemented in a standardised way, which makes it easy to define an equal function for all organ types and to use it for organ update (see Section 2.9 for details).

Table 1   Implemented processes for different organ types

Process	Seed	Root	Bud	Internode	Leaf	Flower	Fruit
Maintenance respiration	-	+	+	+	+	+	+
Growth respiration	+	+	+	+	+	+	+
Photosynthesis	-	-	-	-	+	-	-
Potential growth	-	+	+	+	+	+	+
Actual growth	-	+	+	+	+	+	+

With this technique and the combination of the powerful graph query language integrated into XL [15], it is possible to get information about plant state variables like dry weight of all organs which can be determined by:

```
sum( (* Organ *).getDryWeight() )
```

in an elegant way. The graph query `(* Organ *)` searches all instances of the type `Organ` within the graph and returns them. In a second step the function `getDryWeight()` is called for each object found. Finally, all results are aggregated by the sum function. Most of the functions implemented in the individual module are defined according to this scheme:

```
public float getDryWeight() {
 return sum(
   (* x:Organ, (x.getIndiID()==indiID) *)
   .getDryWeight()
 );
}
```

In case more than one individual is initialized, the condition `(x.getIndiID()==indiID)` makes sure only to output the dry weight of organs of the same individual, i.e., with the matching individual identification number (`indiID` is a constant defined in the parameter file).

Another principle that we use is known from object-oriented programming as "encapsulation" or "information hiding", where information or data are protected from direct access from outside. All conditions for use inside the rules are implemented as functions of organs, e.g., the conditions for a seed to germinate are implemented in the form of a boolean function `isGerminationConditions()` inside the seed module definition, and are used as a simple function call

```
s:Seed, (s.isGerminationConditions()) ==>
    Root(s[indi]) s Bud(1, 1, s[indi]);
```

in the rule definition.

3) http://commons.apache.org
4) http://jscience.org

2.13   Adding a new process to the model

Two main procedures can be applied to add a new process to the FSPM-P. The first procedure involves the linking of a programme which describes the process to be added and which is written in another language. XL being an extension of Java, such a programme could be wrapped using a Java interface allowing the inclusion of libraries (e.g., Apache Commons[3] or JScience[4]), packages and implementations from other languages. Though possible, this is not part of the philosophy of FSPM-P, because other approaches like OpenAlea [45] are much more tailored to conduct "gluing" of heterogeneous models (besides, GroIMP has already provided an http-based interface Open GroIMP, which is used to communicate with OpenAlea).

The second and preferable procedure to integrate new processes into the model is to implement them directly in the FSPM-P code. The object-oriented approach used in FSPM-P facilitates the implementation of a new function. By implementing the new function in the definition of the general organ superclass, it becomes available for all organ types, and then this new function can be adapted or modified in the definition of the concrete organ type if required. For example, growth respiration is always calculated taking the actual growth rate and multiplying the latter with a constant, organ-type specific factor. The general function for growth respiration and its actual implementation for each organ type are stated in the definition of the organ superclass:

```
// growth respiration [g]
public float getGrowthRespiration() {
  return getActualGrowth()
    * ASSIMILATE_LOSS_GR_ORGAN[getOrganType-
      ()];
}
```

In cases where the growth respiration as defined in the organ class does not fit, it can be overwritten in the definition of the specific organ type.

The main temporal resolution of our model is either daily or hourly run mode. To manage different time steps between different processes, e.g., to compute morphological rules each day and light interception at each hour between 6am and 8pm, the user can add conditions for each process when to execute:

```
if(hourOfDay == 12) morphology();
if(hourOfDay >= 6 && hourOfDay <= 20) {
  lightInterception();
}
```

2.14   Visualization

FSPM-P implements a general model, which means that it is not associated with a fixed plant species and thus also not parametrized for a certain species. The parametrization is chosen such that plausible (qualitatively realistic) growth and development will be generated. Figure 3 shows the generated 3D structure at different ages. Additionally, a measurement ruler for visual comparison has been inserted, as well as a black, one square meter large patch as ground which serves for verification of the light model, i.e., to determine the amount of light reaching the ground.

(a)   (b)

(c)   (d)
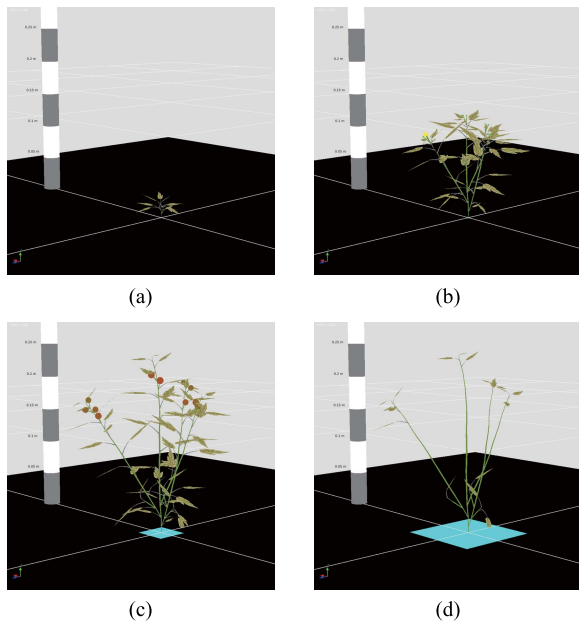
Fig. 3   Generated 3D structure of the FSPM-P model at different development stages. (a) Age 25: juvenile plant; (b) age 50: young plant, first reproduction organs (flowers) occur; (c) age 75: adult plant, fruits at different maturity levels have developed; (d) age 110: terminal stage, fruits have dropped (or been harvested), most basal leaves have been shed due to leaf mortality

3   Results and discussion

3.1   Simulated model output

To monitor and document the dynamics of growth and development processes, a variety of charts have been implemented, e.g., dynamics of organ dry weight and length.

Even without a proper parametrization for a certain crop species, the model has already exhibited general patterns similar to those found in plants, with respect to the phenology of growth stages or stem extension dynamics. Figure 4(a) shows simulated dry weight of leaves as a function of leaf rank. It can be seen that most leaves do not reach their potential dry weight, probably because of the competition for substrate among too many concurrently unfolding leaves while source leaves are still limiting.
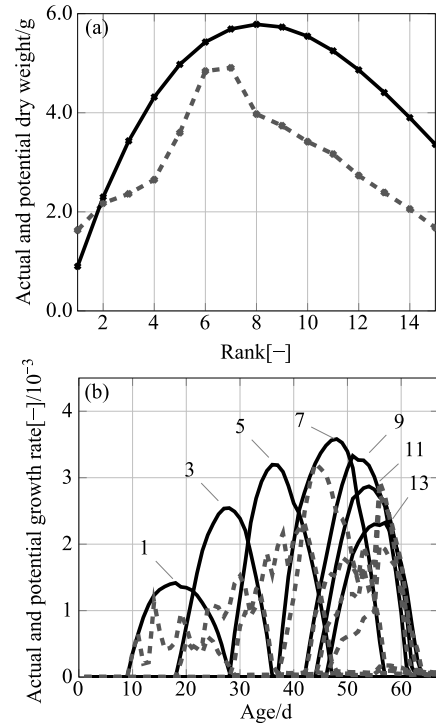

Fig. 4   Simulated model output concerning leaf blades. (a) Final dry weight given as input for the potential growth rate (solid curve) and simulated final dry weight of main leaf blades (dashed curve); (b) potential (solid lines) and actual (dashed lines) growth of main stem leaf blades (rank 1, 3, 5, 7, 9 11 and 13) input

In the current implementation, the final dry weight is rank-dependent for leaves and internodes, while for other organ types it is considered to be equal for each rank. The values used in the FSPM-P are hypothetical, to be subsequently replaced with real measurements. For this prototype, which is a showcase, we took sample data which can be described by a polynomial function.

On the other hand, basal and median leaves nearly reach their potential dry weight: early leaves have little competition with other organs, whereas growth of late leaves is supported by the source strength of many older leaves and (at least before onset of fruit growth (a strong sink)) again experience little competition with other growing organs. The potential and actual growth rate of main stem leaf blades is

shown in Fig. 4(b). It can be seen that the realized growth is almost always smaller than potential growth. Since the model is not specific for a certain crop, this has no further meaning. However, if the model had been parametrized for a species, this could mean that the assimilation rate given by the photosynthesis model is too low (due to insufficient photosynthetic efficiency). As plant growth is almost never reaching its potential but is limited by a shortage of nutrients, water, or light, suboptimal temperatures or pests and diseases, the measured growth rate by definition can not surpass the potential growth rate.

Figure 5(a) demonstrates the simulated dynamics of the carbon assimilation (dry weight) of main stem internodes. According to the parabolic shape of measured final dry weight of internodes (Fig. 5(b)) as a function of rank used as input to calculate potential growth, the simulated final dry weight (Fig. 5(a)) shows the same pattern of internodes with same weight.
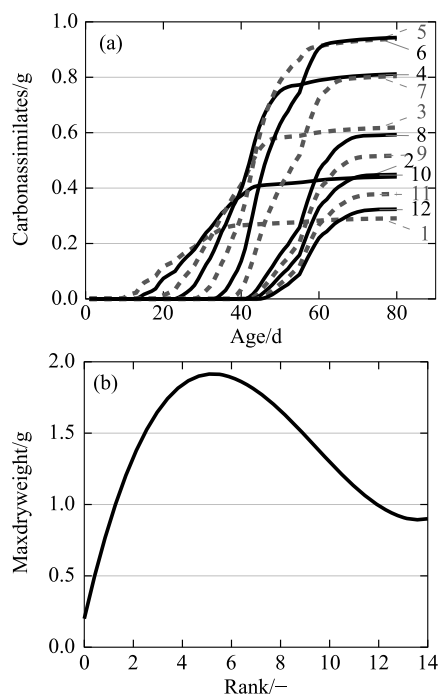
Fig. 5 Comparison of final dry weight of internodes and the measured input function. (a) Simulated dynamics of the carbon assimilation for internodes of rank 1–12; (b) hypothetical "observations" of maximal dry weight as function of rank used as model input

3.2   Importance of a prototype approach

The growing recognition of the FSPM approach, as a logical continuation of the crop modelling tradition [46] (see also the other articles in that special issue on FSPM), necessitates the provision of possibilities for efficient model development as well as for maintenance, support and enhance-

ment. An FSPM, like any other computer programme, can draw substantial benefit and advantage from computer science techniques, mainly software engineering, e.g., object-oriented programming, modularisation, design patterns, software re-usability and basic programming standards [47]. This can enhance both the models themselves and the development process, turning it more structured, efficient, and clearer.

By applying such good practices, models will become easier to understand and better comparable, and submodels can be replaced more easily. Development, combination, implementation, calibration and validation of models can equally benefit from such good practices. The establishment of the best practice in FSPM is a solution for recurring problems, and would rationalise work and enhance productivity as it reduces time for coding, testing and documentation. A predefined and consistent solution like a prototype can also provide standards for testing of parts or the whole model.

A related approach, OpenAlea [45], is a distributed collaborative effort to develop Python libraries and tools that address the needs of current and future work in Plant Architecture modelling. OpenAlea includes modules to analyse, visualize and model the functioning and growth of plant architecture. However, the difference is that OpenAlea essentially links different programmes (potentially written in different languages and exhibiting different compilation states: dll, source code, etc.), whereas our approach is a core FSPM that runs a priori, and that has already included the main functional elements (light interception, photosynthesis, etc.), in the same programming environment and language (GroIMP and XL).

The GreenLab approach [11] is comparable to the present model, as it provides a fully runnable model that can be parametrized for different species. However, its source function being based on radiation use efficiency and lacking internal transport, it falls short of the generality which we consider as necessary for an extensible FSPM. In this respect, it is closer related to CANON [48], in which a composite design pattern was implemented at the phytomer level for use in a FSPM.

FSPM-P can be seen as the first step to a general FSPM which, in its first version, is presented as a conceptual model including a user manual with explanations about experiment set-up, measurement protocols, data processing, model description and parametrizations, and the model itself.

In terms of a model classification, e.g., the pedigree of "de Wit" models [1, 49], our approach is not strictly comparable as it explicitly considers structure in 3D. However, it can be classified according to the (fairly large number of) processes

it describes and the level of detail it provides, as a potential production model working at the physiological level of detail [1]: it neither consider the effect of limitations of water and nutrients (for this, an extension to a root-soil interface model would be necessary) nor the effect of pests and diseases on crop production, yet it considers three of the four main ecophysiological processes listed by Goudriaan and van Laar [1] — carbon assimilation, plant development, and respiration, disregarding plant transpiration. However, since the LEAFC3 model, which is provided in our library of photosynthesis models, also computes potential evapotranspiration, an extension to cover plant transpiration is within reach.

3.3   Possible further application areas

An important feature of FSPM-P is the fact that it already constitutes a running model, which can thus be used straight away. Its use as a departure point for developing a dedicated FSPM of a certain crop is thus obvious.

A further, immediate application is its utilization in teaching and for presentations in the plant sciences where it is often necessary to demonstrate a process in a general way. Crop models without a consideration of plant architecture like LINTUL [50] or SUCROS [51] have been successfully used for teaching purposes [1]. Our approach currently permits the modelling of both individual plants and plant stands (canopies), where the latter are potentially consisting of a mixture of two or more different species. The possibility to model mixed stands makes it suitable for application in intercropping. The more or less concurrent cultivation of two crops in the same field is a very important technique, e.g., in Chinese agriculture. While it seems to be more resource-use efficient than conventional mono-cropping, it also poses substantial challenges with respect to understanding the underlying mechanisms. The most common advantage of intercropping is the production of greater yield on a given piece of land [52]. Furthermore, Ouma [53] also took risk minimization and reduction of soil erosion into consideration, and increased food security as advantages of intercropping. Both publications illustrate clearly the high potential of intercropping as sustainable alternative. An FSPM-P adapted for an intercropping system could be used to investigate and analyse competitive and facilitative relationships between the crop species involved in detail, both above-ground and below-ground, and to elucidate dynamic interactions in space and time at the level of plant organs (e.g., leaves and roots). Calculations done using the FSPM-P would thus help to explain eco-efficiencies in field experiments on the basis of causal

ecological mechanisms, and could then be used to explore opportunities for improved intercrop performance by modified system design (species choice, sowing date, planting pattern, irrigation and fertilisation).

Modelling interaction between root systems of different species with each other and with the soil, is an essential element in the investigation of intercropping systems. Modelling the soil requires a discretization of space into cubes with properties such as nutrition, resistance, or water status, and of the dynamics of water and nutrient movement. We have conducted a preliminary implementation study [54], in which we have created a simple yet modular root-soil interaction model. Such a model could thus be used as a generic module in a larger intercropping model system.

3.4   Possible extensions

The range of possible extensions is quite diversified and could encompass the following:

• More detailed carbon storage and transport concepts The central carbon pool concept is an extreme simplification and biologically not well founded. A transport-based concept with local organ pools would be more realistic. The latter is a concept which is very relevant for our approach. Once this extension is implemented, our model could be used to test hypotheses from plant physiology, such as central versus local pools, ranges and modes of transport (e.g., diffusion, convection and active transport).

• Extension to a general tree model   The current version of FSPM-P is mainly adapted to small plants with a vegetation period of less than one year. However, it would be interesting and not difficult to extend and change FSPM-P to simulate perennial and polycarpic trees.

• Component-based plant model   The ultimate objective of this project is to design a user-friendly general FSPM with generic modules representing functions and processes, plant organs, architectural characteristics or communication and transport which can be used as components and simply combined to a model using a kind of graphical editor. Such an approach with independently developed, verified and reusable components can further facilitate the comparison and exchange of submodels as well as their evaluation and standardization.

• Calculation of spectral light   The use of the GPUFlux ray tracer provides several opportunities to not only simulate light distribution over the full spectrum of light, but also allow to calculate, e.g., relations between red and far red light. In combination simulations of artificial light sources

with specific spectral power distributions and physical light distributions, common light conditions, e.g., found in greenhouses or climate chambers, can be reproduced and further used for functional-structural plant modelling.

4 Outlook and conclusions

The model presented here is the first step towards establishing a general model with standardised modules, processes and communication structure, which enables a clear model design, and is easy to parametrize (see Appendix B), understand and extend.

This systematic approach provides all the necessary infrastructure and documentation to develop efficient FSPMs based on their own measurements for different target groups (with or without knowledge of programming or modelling) and could also be useful for professional FSPM developers as a basic framework.

FSPM-P is nevertheless open for arbitrary extensions by rule-based coding in the language XL, thus its application is not restricted to a predefined range of parameter values or to a preselected portfolio of shapes or processes. Finally, a prototype like the one presented here will facilitate communication between modeller, programmer and experimentator, which can be mutually beneficial and helpful in establishing FSPM as a tool for research, development and education in the plant and crop sciences.

Appendix

Appendix A List of abbreviations

Abbreviation	Description	Unit
AG	actual growth	g
AGR	actual growth rate	g/s
AP	temporary assimilate pool	g
CPU	central processing unit	
CSV	comma-separated values	
FSPM	functional-structural plant model	
FSPM-P	FSPM-Prototype	
GPU	graphics processing unit	
GR	growth respiration	
LD	local demand	g/s

Abbreviation	Description	Unit
L-systems	Lindenmayer-systems	
MR	maintenance respiration	
ODE	ordinary differential equation	
PAR	photosynthetically active radiation	Wm^{-2}
PPFD	photosynthetic photon flux density	$\mu mol P/m^{-2}s$
PGR	potential growth rate	g/s
RSS	relative sink strength	
XL	eXtended L-System modelling language	

Appendix B External parameter files

The external parameter files are an elegant way to easily (re-)configure the FSPM-P. For example, for scenario tests, each configuration is stored in an individual file, where the user can switch between by changing only one entry.

The parameter files follow the syntax of common property files, which are widely used to configure software. Property files are simple text files, which can have maximal one entry per line. An entry consists of a key / identifier followed by an equals sign and the actual value of this entry:

$$< key > = \; value.$$

The *key* is a string used to identify this entry. We use the same name here as used later in the model code to make it traceable and transparent. In the current implementation, *value* can be one of the following types: *String*, *Integer*, *Double*, *Boolean*, or an array of one of them.

Below you can find a part of the *scenario.ini* file, which is used to configure the whole configuration for one specific scenario. Here it can be defined, e.g., which climate file, which photosynthesis model has to be used, or the start day for the simulation.

```
// species parameter
SPECIES_PARAMETER_FILE = speciesParameters.ini
// climate data:  Meteostation Haarweg 2008
CLIMATE_DATA_FILE = climateHaarweg2008Daily.csv
// number of values in CLIMATE_DATA_FILE
CLIMATE_DATA_VALUES = 366

// environment data
ENVIRONMENT_DATA_FILE = environment.ini

// debug mode
DEBUG_MODE = true

// show benchmark informations each step
BENCHMARK = false

// activate data logging
```

```
USE_LOG_FILE = false

// determinates the run modi of the model
DAILY_RUN_MODE = true

// day of the year; model starts at:  April 1st
START_DAY = 121

// select the photosynthesis model
(LEAFC3N2010=0,
//LIETHPASIAN = 1, KIMLIETH = 2, THORNLEY = 3,
//THORNLEYN = 4, MARSHALLBISCOE = 5,
//JOHNSONTHORNLEY = 6, HOST = 7, BALDOCCHI = 8)
PHOTOSYNTHESIS_MODEL = 8

// light model (CPU = 0, GPU = 1)
LIGHT_MODEL = 1
...
```

References

1. Goudriaan J, Van Laar H H. Modelling Potential Crop Growth Processes: Textbook with Exercises. Dordrecht: Kluwer Academic Publishers, 1994

2. Lopez G, Favreau R P, Smith C, Costes E, Prusinkiewicz P, DeJong T M. Integrating simulation of architectural development and source-sink behaviour of peach trees by incorporating Markov chains and physiological organ function submodels into L-PEACH. Functional Plant Biology, 2008, 35(10): 761–771

3. Allen M T, Prusinkiewicz P, DeJong T M. Using L-systems for modeling source-sink interactions, architecture and physiology of growing trees: the L-PEACH model. New Phytologist, 2005, 166(3): 869–880

4. Xu L F, Henke M, Zhu J, Kurth W, Buck-Sorlin G H. A rule-based functional-structural model of rice considering source and sink functions. In: Proceedings of the 3rd International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications. 2009, 245–252

5. Buck-Sorlin G H, de Visser P H B, Sarlikioti V, Burema B S, Heuvelink E, Marcelis L F M, van der Heijden G W A M, Vos J. SIMPLER: an FSPM coupling shoot production, human interaction with the structure, morphogenesis, photosynthesis and light environment in cut-Rose. In: Proceedings of the 6th International Workshop on Functional-Structural Plant Models. 2010, 222–224

6. Groer C, Kniemeyer O, Hemmerling R, Kurth W, Becker H, Buck-Sorlin G H. A dynamic 3D model of rape (Brassica napus L.) computing yield components under variable nitrogen fertilization regimes. In: Proceedings of the 5th International Workshop on Functional-Structural Plant Models. 2007

7. Buck-Sorlin G H, Kniemeyer O, Kurth W. Barley morphology, genetics and hormonal regulation of internode elongation modelled by a relational growth grammar. New Phytologist, 2005, 166(3): 859–867

8. Buck-Sorlin G H, Kniemeyer O, Kurth W. A grammar-based model of barley including genetic control and metabolic networks. In: Proceedings of Functional-Structural Plant Modelling in Crop Production. 2007, 243–252

9. Buck-Sorlin G H, Hemmerling R, Kniemeyer O, Burema B, Kurth W. A rule-based model of barley morphogenesis, with special respect to shading and gibberellic acid signal transduction. Annals of Botany, 2008, 101(8): 1109–1123

10. Barczi J F, Rey H, Caraglio Y, Reffye d P, Barthélémy D, Dong Q X, Fourcaud T. AmapSim: a structural whole-plant simulator based on botanical knowledge and designed to host external functional models. Annals of Botany, 2008, 101(8): 1125–1138

11. Hu B G, Reffye P D, Zhao X, Yan H P, Kang M Z. GreenLab: a new methodology towards plant functional-structural model — structural aspect. In: Hu B, Jaeger M, eds. Plant Growth Modeling and Applications. Beijing: TsingHuo University Press and Springer, 2003, 21–35

12. Letort V. Analyse multi-échelle des relations source-puits dans les modèles de développement et croissance des plantes pour l'identification paramétrique. Cas du modèle GreenLab. Pissertation for the Doctoral Degree. Châtenay-Malabry: École Centrale Paris, 2008

13. Breckling B. An individual based model for the study of pattern and process in plant ecology: an application of object oriented programming. EcoSys, 1996, 4: 241–254

14. Perttunen J, Sievänen R, Nikinmaa E, Salminen H, Saarenmaa H, Väkevä J. LIGNUM: A tree model based on simple structural units. Annals of Botany, 1996, 77(1): 87–98

15. Kniemeyer O. Design and implementation of a graph grammar based language for functional-structural plant modelling. Dissertation for the Doctoral Degree. Cottbus: Brandenburg University of Technology, 2008

16. Kurth W. Morphological models of plant growth. Possibilities and ecological relevance. Ecological Modelling, 1994, 75: 299–308

17. Prusinkiewicz P, Lindenmayer A. The Algorithmic Beauty of Plants. New York: Springer Science & Business Media, 2012

18. Hemmerling R. Extending the programming language XL to combine graph structures with ordinary differential equations. Dissertation for the Doctoral Degree. Göttingen: University of Göttingen, 2012

19. Hemmerling R, Kniemeyer O, Lanwert D, Kurth W, Buck-Sorlin G H. The rule-based language XL and the modelling environment GroIMP illustrated with simulated tree competition. Functional Plant Biology, 2008, 35(9/10): 739–750

20. Van Antwerpen D G. Unbiased physically based rendering on the GPU. Dissertation for the Master Degree. Delft: Delft University of Technology, 2011

21. Veach E. Robust Monte Carlo Methods for Light Transport Simulation. Dissertation for the Doctoral Degree. Palo Alto: Stanford University, 1998

22. Buck-Sorlin G H, Hemmerling R, Vos J, de Visser P H. Modelling of spatial light distribution in the greenhouse: Description of the model. In: Proceedings of the 3rd International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications. 2009, 79–86

23. Evers J B, Vos J, Yin X, Romero P, Van Der Putten P E L, Struik P C. Simulation of wheat growth and development based on organ-level photosynthesis and assimilate allocation. Journal of Experimental Botany, 2010, 61(8): 2203–2216

24. Preetham A J, Shirley P, Smits B. A practical analytic model for daylight. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. 1999, 91–100

25. Gijzen H. Development of a simulation model for transpiration and water uptake and an integral growth model. AB-DLO Report 18. 1994

26. Nikolov N T, Massman W J, Schoettle A W. Coupling biochemical and biophysical processes at the leaf level: an equilibrium photosynthesis model for leaves of C3 plants. Ecological Modelling, 1995, 80: 205–235

27. Müller J, Wernecke P, Diepenbrock W. LEAFC3-N: a nitrogen-sensitive extension of the CO2 and H2O gas exchange model LEAFC3 parameterised and tested for winter wheat (Triticum aestivum L.). Ecological Modelling, 2005, 183: 183–210

28. Müller J, Braune H, Diepenbrock W. Photosynthesis-stomatal conductance model LEAFC3-N: specification for barley, generalised nitrogen relations, and aspects of model application. Functional Plant Biology, 2008, 35: 797–810

29. Baldocchi D. An analytical solution for coupled leaf photosynthesis and stomatal conductance models. Tree Physiology, 1994, 14: 1069–1079

30. Kim S H, Lieth J H. A coupled model of photosynthesis, stomatal conductance and transpiration for a rose leaf (Rosa hybrida L.). Annals of Botany, 2003, 91(7): 771–781

31. Lieth J H, Pasian C C. A simulation model for the growth and development of flowering rose shoots. Scientia Horticulturae, 1991, 46: 109–128

32. Thornley J H M. A model to describe the partitioning of photosynthate during vegetative plant growth. Annals of Botany, 1969, 33: 419–430

33. Thornley J H M. Dynamic model of leaf photosynthesis with acclimation to light and nitrogen. Annals of Botany, 1998, 81(3): 421–430

34. Johnson I R, Thornley J H M. Dynamic model of the response of a vegetative grass crop to light, temperature and nitrogen. Plant, Cell and Environment, 1985, 8(7): 485–499

35. Marshall B, Biscoe P V. A model for C3 leaves describing the dependence of net photosynthesis on irradiance I. Derivation. Journal of Experimental Botany, 1980, 31(1): 29–39

36. Marshall B, Biscoe P V. A model for C3 leaves describing the dependence of net photosynthesis on irradiance II. Application to the analysis of flag leaf photosynthesis. Journal of Experimental Botany, 1980, 31(1): 41–48

37. Rauscher H M, Isebrands J G, Host G E, Dickson R E, Dickmann D I, Crow T R, Michael D A. ECOPHYS: an ecophysiological growth process model for juvenile poplar. Tree Physiology, 1990, 7: 255–281

38. Yin X Y, Goudriaan J, Lantinga E A, Vos J, Spiertz H J. A flexible sigmoid function of determinate growth. Annals of Botany, 2003, 91(3): 361–371

39. Richards F J. A flexible growth function for empirical use. Journal of Experimental Botany, 1959, 29(10): 290–300

40. Thornley J H M. Growth, maintenance and respiration: a reinterpretation. Annals of Botany, 1977, 41(6): 1191–1203

41. Bertin N, Gary C. Évaluation d'un modèle dynamique de croissance et de développement de la tomate (Lycopersicon esculentum Mill), TOMGRO, pour différents niveaux d'offre et de demande en assimilats. Agronomie, 1993, 13: 395–405

42. Marcelis L F M. A simulation model for dry matter partitioning in cucumber. Annals of Botany, 1994, 74(1): 43–52

43. Marcelis L F M. Sink strength as a determinant of dry matter partitioning in the whole plant. Journal of Experimental Botany, 1996, 47: 1281–1291

44. Qi R, Ma Y T, Hu B G, de Reffye P, Cournède P H. Optimization of source-sink dynamics in plant growth for ideotype breeding: a case study on maize. Computers and Electronics in Agriculture, 2010, 71(1): 96–105

45. Pradal C, Dufour-Kowalski S, Boudon F, Fournier C, Godin C. OpenAlea: a visual programming and component-based software platform for plant modelling. Functional Plant Biology, 2008, 35(10): 751–760

46. Vos J, Evers J B, Buck-Sorlin G H, Andrieu B, Chelle M, de Visser P H B. Functional-structural plant modelling: a new versatile tool in crop science. Journal of Experimental Botany, 2010, 61(8): 2101–2115

47. Wilson G V. Where's the real bottleneck in scientific computing? American Scientist, 2006, 94(1): 5–6

48. McMaster G S, Hargreaves J N G. CANON in D(esign): composing scales of plant canopies from phytomers to whole-plants using the composite design pattern. NJAS - Wageningen Journal of Life Sciences, 2009, 57(1): 39–51

49. Bouman B A M, Keulen v H, Laar v H H, Rabbinge R. The 'school of de Wit' crop growth simulation models: A pedigree and historical overview. Agricultural Systems, 1996, 52(2): 171–198

50. Spitters C J T. Crop growth models: their usefulness and limitations. ISHS Acta Horticulturae 267: VI Symposium on the Timing of Field Production of Vegetables. 1990, 349–368

51. Van Keulen H, Penning de Vries F W T, Drees E M. A summary model for crop growth. In: Penning de Vries F W T, van Laar H H, eds. Simulation of plant growth and crop production, Wageningen: Centre for Aqricultural Publishing and Documentation, 1982

52. Lithourgidis A S, Dordas C A, Damalas C A, Vlachostergios D N. Annual intercrops: an alternative pathway for sustainable agriculture. Australian Journal of Crop Science, 2011, 5(4): 396–410

53. Ouma G P J. Sustainable horticultural crop production through intercropping: the case of fruits and vegetable crops: a review. Agriculture and Biology Journal of North America, 2010, 1(5): 1098–1105

54. Henke M, Sarlikioti V, Kurth W, Buck-Sorlin G H, Pagès L. Exploring root developmental plasticity to nitrogen with a three-dimensional architectural model. Plant and Soil, 2014, 385(1): 49–62

Michael Henke received his Diploma degree in Computer Science from the Brandenburg University of Technology Cottbus, Germany, and his PhD in Biometrics and Forest Growth at the University of Göttingen, Germany in 2007. From 2009 to 2010, he was a visiting scholar at Zhejiang University, China. Currently, he is an assistant lecturer at Brandenburg University of Technology Cottbus and University of Göttingen, Germany. His research interest is functional-structural plant modelling.

Winfried Kurth received his Diploma degree in Mathematics, and PhD in Theoretical Computer Science from Clausthal University of Technology, Germany.Subsequently, He was a junior researcher at the Universities of Göttingen and Bayreuth. From 2001 to 2008, he was a professor in Practical Computer Science and Graphics Systems at Cottbus University of Technology, Germany.Since 2008, he is a professor in Computer Graphics and Ecological Informatics at University of Göttingen, Germany. His research fields include rule-based languages, representation of 3D data, functional-structuralplant models, and simulation.

Gerhard H. Buck-Sorlin received his Diploma degree in Biology at the University of Göttingen. Germany, and his PhD in Biology at the University of Wales in Bangor, UK in 1997. Subsequently, he worked as a junior postdoctoral scientist at the Institute of Plant Genetics and Crop Plant Research in Gatersleben, Germany. He was a senior scientist and lecturer at Cottbus University of Technology, Germany from 2002 to 2007, and a guest professor at the Zhejiang University, China from 2005 to 2009. Since 2011, he is a professor in Fruit Tree Culture and Modelling at Agrocampus Ouest, Centre d'Angers, France. His research fields include ecophysiology of crop plants, and functional-structural plant modelling.