Environmental Modelling & Software 62 (2014) 385–398

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Plant Modelling Framework: Software for building and running crop models on the APSIM platform☆

Hamish E. Brown [a, *], Neil I. Huth [b], Dean P. Holzworth [b], Edmar I. Teixeira [a], Rob F. Zyskowski [a], John N.G. Hargreaves [b], Derrick J. Moot [c]

[a] The New Zealand Institute for Plant & Food Research Limited, Private Bag 4604, Christchurch, New Zealand
[b] CSIRO Ecosystem Sciences/Sustainable Agriculture Flagship, PO Box 102, 4350 Toowoomba, Australia
[c] Faculty of Agriculture and Life Sciences, P.O. Box 85084, Lincoln University, 7647 Canterbury, New Zealand

ARTICLE INFO

Article history:
Received 30 September 2013
Received in revised form
1 July 2014
Accepted 3 September 2014
Available online 1 October 2014

Keywords:
Canopy dynamics
Biomass and nitrogen partitioning
Integrated design environment
Phenological and morphological development
Reusable organ and function classes

ABSTRACT

The Plant Modelling Framework (PMF) is a software framework for creating models that represent the plant components of farm system models in the agricultural production system simulator (APSIM). It is the next step in the evolution of generic crop templates for APSIM, building on software and science lessons from past versions and capitalising on new software approaches. The PMF contains a top-level Plant class that provides an interface with the APSIM model environment and controls the other classes in the plant model. Other classes include mid-level Organ, Phenology, Structure and Arbitrator classes that represent specific elements or processes of the crop and sub-classes that the mid-level classes use to represent repeated data structures. It also contains low-level Function classes which represent generic mathematical, logical, procedural or reference code and provide values to the processes carried out by mid-level classes. A plant configuration file specifies which mid-level and Function classes are to be included and how they are to be arranged and parameterised to represent a particular crop model. The PMF has an integrated design environment to allow plant models to be created visually. The aims of the PMF are to maximise code reuse and allow flexibility in the structure of models. Four examples are included to demonstrate the flexibility of application of the PMF; 1. Slurp, a simple model of the water use of a static crop, 2. Oat, an annual grain crop model with detailed growth, development and resource use processes, 3. Lucerne, perennial forage model with detailed growth, development and resource use processes, 4. Wheat, another detailed annual crop model constructed using an alternative set of organ and process classes. These examples show the PMF can be used to develop models of different complexities and allows flexibility in the approach for implementing crop physiology concepts into model set up.

Availability

The Plant Modelling Framework source code is freely available for non commercial use and can be viewed at http://apsrunet.apsim.info/websvn/listing.php?repname=apsim&path=/trunk/ then clicking on the "Model" then "Plant2" folders. Note that the PMF (called Plant2 in internal documentation) does not stand alone and users will need to download the Agricultural Production Systems Simulator (http://www.apsim.info/Products/Downloads.aspx) to build and use PMF models.

1. Introduction

A key purpose of APSIM is to simulate realistic long-term dynamics in agricultural simulations (Holzworth et al., 2014; Keating et al., 2003). To do this, a range of arable, pasture, vegetable, tree, bush, vine and weed models are required to represent different kinds of plant communities and their contributions to the water and nutrient balance of agricultural land. However, the development and maintenance of many models requires considerable time and financial commitment. This problem is relevant to all agricultural systems modelling platforms that provide the capacity to simulate different crop types (Brisson et al., 2003; Jones et al., 2003; Stockle et al., 2003). Generic crop templates have been developed to address the problem (van Keulen et al., 1982; Penning de Vries

☆ Thematic Issue on Agricultural Systems Modeling & Software.

* Corresponding author. Tel.: +64 3 325 9394.
E-mail address: hamish.brown@plantandfood.co.nz (H.E. Brown).

http://dx.doi.org/10.1016/j.envsoft.2014.09.005

et al., 1989). They are based on the hypothesis that crop models can be constructed from a generic set of software classes that are then assembled and parameterised differently to represent the physiology of different crops. From this, the idea of process oriented programming was developed where sub routines represent a process defined as "a series of events, which drive the dynamics of the system in response to system attributes and environmental conditions" (Wang et al., 2003; Wang and Engel, 2000). A crop was defined as a system with a set of components such as phenology, organ genesis and biomass production. Processes are closely related to a specific system component and result in the change of the components state variables. Efforts have been made by a number of groups to increase flexibility in the way that generic crop templates are set up to make models more adaptable to different requirements. The DSSAT group developed the CROPGRO template (Boote et al., 1998) in which crop coefficients are set in a 'species file' to configure the model as a particular crop type. CROPGRO has been applied to a large number of crops but has forced developers into a fixed structure and numerous changes have been required to the code base to adapt it to different crop types. The APES simulator (Donatelli et al., 2010) provides a range of pre-determined crop model component options and parameters that the user selects through a graphical user interface (GUI). Similarly CROSPAL (Adam et al., 2010) provides a framework which combines expert knowledge and libraries containing crop modelling with a GUI that allows crop models to be assembled using combinations of different modelling approaches, However, to date CROSPAL based plant models have not been incorporated into a full farm systems model.

Specific to the APSIM model, Wang et al. (2003) developed the generic crop model template (GCROP). This consisted of a set of component and function classes in a crop process library (source code) and configuration files. The configuration file informed the APSIM model of what classes to assemble and what their parameter values were. This meant model developers could focus on determining crop parameter values and evaluating crop models without the need to write model code. Reusing the same code for each model also meant maintenance was simplified because the code base was smaller and changes did not have to be repeated in multiple code sets whenever a fix or enhancement was implemented. In 2003 a new template was developed for APSIM which was derived from the ideas and structure of GCROP and the generic legume model (Robertson et al., 2002) implemented in C++ and named PLANT in recognition of the application of the template to communities of plant species other that those defined as crops.

From the 41 plant models currently in APSIM (Holzworth et al., 2014), 26 are implemented using the PLANT or GCROP templates including cereals (Keating et al., 2001; Manschadi et al., 2006; Peake et al., 2008), legumes (Robertson and Carberry, 1998; Robertson et al., 2002; Turpin et al., 2003), horticultural crops (Robertson et al., 2002), vines (Huth et al., 2009; Robertson et al., 2005), pastures (Dolling et al., 2005; Probert et al., 1998b; Verburg et al., 2007) and weeds (Thornby et al., 2006; Whish et al., 2002). However, over time a number of limitations have been exposed in the PLANT template approach. In particular, it has a limited set of functionality available to model developers and creating new functions is difficult due to the structure and implementation of the template. This has forced models into a fixed structure. However, different researchers have different, but equally acceptable, ideas about how reality should be abstracted. For instance, if data describing environmental responses are limited and the intended scope of model application is limited to specific situations (e.g. the study of yield under irrigated conditions at a single location), a simple model might be preferred. If the physiology of crop has been intensively studied and a model is to be used in a number of different situations (e.g studying yield, water and nitrogen balance in a range of locations and management systems), a more detailed crop model will be preferred. Both approaches are legitimate in certain circumstances. As a result of the prescribed form of the generic template, crop modellers who want to take different approaches have not used it. Instead they have integrated alternative models into APSIM, resulting in contrasting code bases for crop models and a maintenance burden for the software team.

New developments in the software industry (David et al., 2013), as well as lessons learned over the last 10 years from building models in the GCROP and PLANT templates (Holzworth and Huth, 2009b; Holzworth et al., 2010), have made an update to the generic crop template possible. Newer computer programming languages (e.g. C#) have the ability to inspect source code at runtime, extracting metadata about the code. This ability, called reflection or introspection (Rahman et al., 2004), can be used by a model framework to locate required functionality (classes) and determine their data requirements (inputs). This means that classes can be developed independently and models can be constructed at run time from executable mark-up language (XML) files that specify how to assemble and parameterise classes. The updated template is called the Plant Modelling Framework (PMF). This has been in development over the past 3 years aiming to achieve a number of key design goals:

- Enable models to be established at different levels of complexity.
- Enable code reuse and minimise the amount of code to maintain.
- Externalise the structure and parameterisation of a crop model.
- Provide a framework that enables the easy inclusion of new organ, process and function class alternatives.

The aim of this paper is to describe the PMF both conceptually and technically as a modular framework for building crop models and provide a set of example models of contrasting complexity to demonstrate its application and flexibility. One of these models is fully validated and previously implemented in APSIM (Wheat), one is a simple model also previously implemented in APSIM (Slurp) and two are alternatives to current APSIM models and still under development (Oat and Lucerne). This paper is not intending to provide a full description of these example models but use them to demonstrate different ways that science is translated into software in the PMF.

2. Description of the Plant Modelling Framework

The PMF classes are abstracted at the plant or sub-plant level. However, an instance of PMF represents a community of identical plants so the basic units for class properties are expressed on an area basis (m^{-2}). Italics are used when referring to the names of specific classes, properties, events and values within the PMF software and names that refer specifically to the examples shown in Fig. 1 are given in single quotes. There are three main types of classes in the PMF (Fig. 2):

- **Top-level *Plant* class** which provides an interface between instances of the crop model and the other models in the APSIM environment such as soil models and weather data. It also serves as an interface with the other PMF classes.
- **Mid-level classes.**
  - **Organ classes** which contain properties including; current status, supplies and demands of biomass (dry mass,*DM*, and nitrogen,*N*, mass); dimension (e.g. *Height* and *LeafAreaIndex*) and the number of constituents (e.g. *GrainNumber*) within

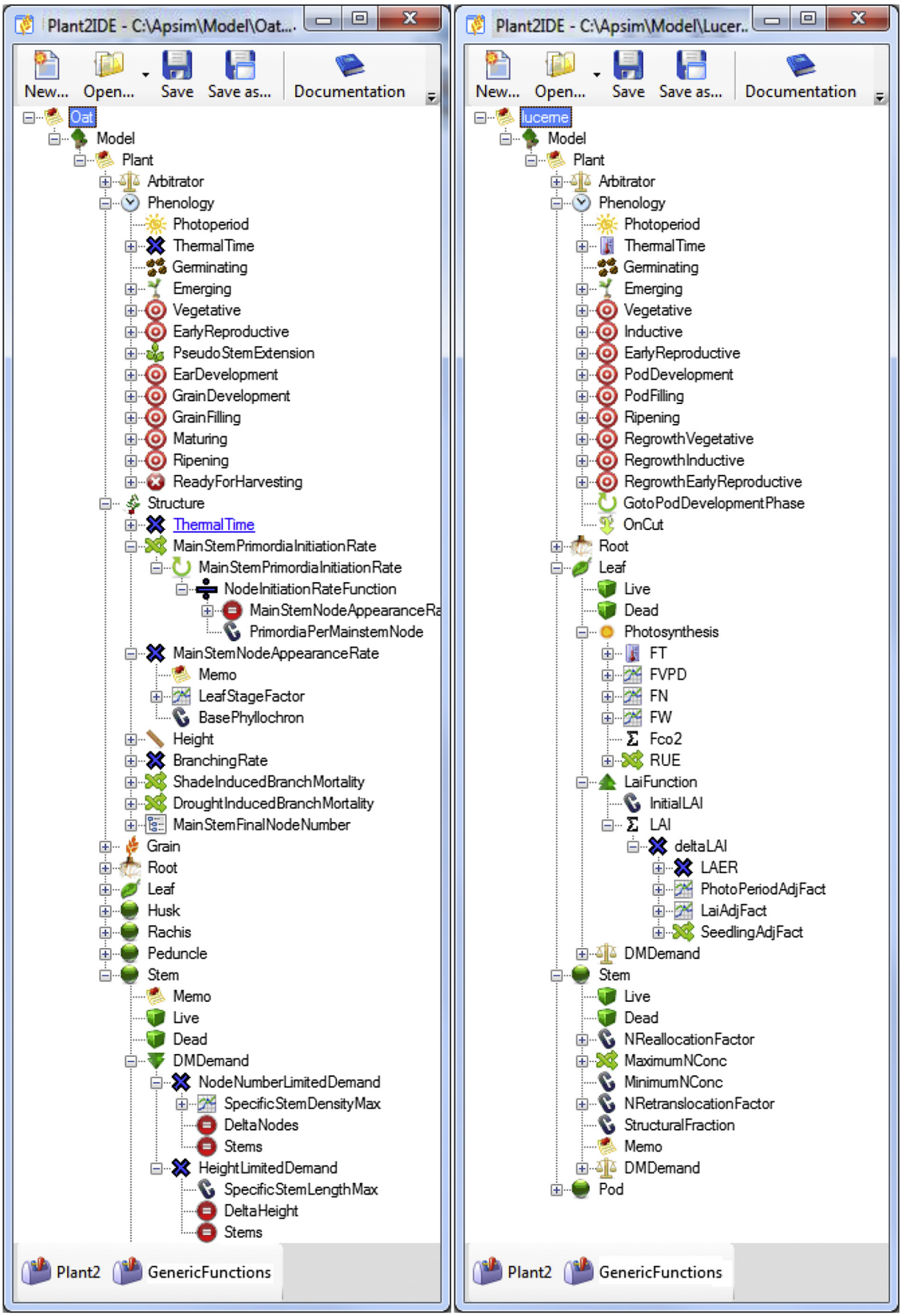

Fig. 1. Screen shots showing the basic structure of Oat (Left) and Lucerne (right) configuration files with selected detail expanded.

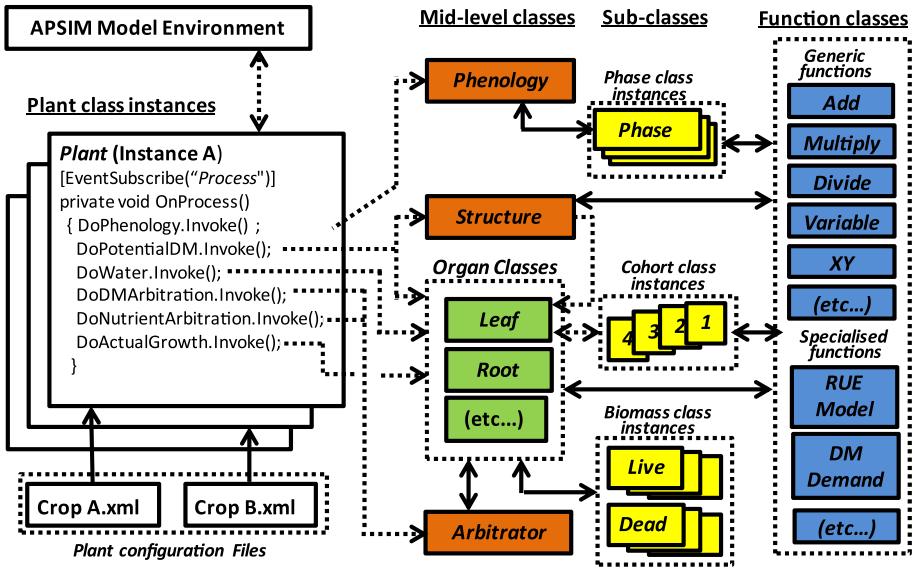

Fig. 2. Schematic of the Plant Modelling Framework. Class types in the PMF source code include: the top level plant class which interfaces with the model environment, organ classes (green shaded boxes), process classes (orange boxes), sub-classes (yellow boxes) that contain repeatable sub-sets of specific aspects of organ and process classes, function classes (blue shading) that contain mathematical, procedural, logical or reference code to provide values to the other classes. Arrows show the direction of communications. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

that organ. These include *Root*, *Leaf*, *Reproductive* and *Generic* organ classes.

- **Process classes** that orchestrate activity at the plant level and provide cues or instructions to organ classes. These include the arbitration of resource allocation to organs (*Arbitrator* class) and the phasic (*Phenology* class) and morphological (*Structure* class) development of the crop.

- **Sub-classes** that organ and process classes delegate to for processes and/or data structures that repeat the same pattern. Multiple instances of a sub-class are used to represent each repetition (e.g. live and dead biomass, phases of development, cohorts of leaves).

- **Low-level function classes** contain generic mathematical, logical, reference or procedural code to return values to be used in mid-level class calculations.

Processes are programmed in a general way within the source code of mid-level classes. An example is shown in Fig. 3 for the *Structure* class. This class has a property of *MainStemNodeNo* who's value is calculated daily in the *DoPotentialDM* event method. The values for *ThermalTime* and *MSNAppRate* which are used in this calculation are links to *Function* classes rather than simple floating point declarations. The PMF infrastructure will look for a *Function* with the same name as the declaration e.g. *ThermalTime* and *MainStemNodeAppearanceRate*. This allows any type of function to be selected providing the names match. Therefore, the model developer can chose different functions to provide values to the source code depending on how they want to represent it in their model. There are a range of broadly generic function classes that are used to implement a specific algorithm that can be used by many higher level classes (Table 1). There are also a number of specialised function classes that perform a single function, such as calculating an organ's *DMdemand*, but are still generic as they can be re-used by any organ and any plant model. Function classes can also take the values they require from other function classes. Therefore, functions can be assembled in many different ways to derive values for the processes contained in mid-level classes.

The type, arrangement and parameterisation of classes to be included in a model is specified by a plant configuration file (executable mark up language) and the PMF has an Integrated Development Environment (IDE) to allow plant models to be constructed visually (Fig. 1). An example of how functions can be assembled to deliver values can be seen in Fig. 1 where the Oat 'Stem' *Organ* class calculates its 'DMDemand' using a *Minimum* function ('DMDemand'). This gets its values from two *Multiply* functions ('NodeNumberLimitedDemand' and 'HeightLimitedDemand') which in turn get their values from *LinearInterpolation* ('SpecificStemDensityMax'), *Constant* ('SpecificStemLengthMax') and *VariableReference* functions ('DeltaNodes', 'DeltaHeight' and

```
[Description("Keeps Track of Plants Structural Development")]
public class Structure
{
    //Class properties
    [Output]
    [Description("Number of mainstem nodes appeared")]
    public double MainStemNodeNo { get; set; }

    //Function Class Links
    [Link(NamePath = "ThermalTime")]
    public Function ThermalTime = null;
    [Link(NamePath = "MainStemNodeAppearanceRate")]
    public Function MSNAppRate = null;

    //Class processes methods
    [EventHandler]
    public void DoPotentialDM()
    {
        MainStemNodeNo += ThermalTime.Value / MSNAppRate.Value;
    }
}
```

Fig. 3. Example of selected source code fragments to demonstrate how mid level classes (the Structure class in this example) are structured to obtain the values they need for their calculations.

Table 1
Examples of function classes within the Plant Modelling Framework. Icons correspond with functions shown in Fig. 1.

Function type	Function name	Return value
Mathematical	Add	Sum of all child class values
	Subtract	First child class value less all others
	Multiply	Product of all child class values
	Divide	The 1st child class value divided by the 2nd
	Exponential	*Y* given *x* variable and a, b, c coefficients
	Sigmoid	*Y* given *x* variable and a, b, c coefficients
	Power	*Y* given *x* variable and exponent coefficient
	Expression	Parse and solve any expression for given *x* variable.
	LinearInterpolation	*Y* given *xy* coordinates and *x* variable
	Constant	Single fixed value
Logical	Maximum	Maximum of all child function values
	Minimum	Minimum of all child function values
	LessThan	If variable reference < criteria return child 1 value, else return child 2 value
	GreaterThan	If variable reference > criteria return child 1 value, else return child 2 value
Reference	Variable	Value of specific PMF class property
	ExternalVariable	Value of specific property in APSIM environment
	DayLength	Day length from latitude and day or year
Procedural	OnEvent	Value 1 prior to event and Value 2 after event
	PhaseLookup	Value of child function associated with each phenological phase
	Age	Days since sowing
	Accumulate	Child values accumulated daily

'Stems'). While functions that deliver values to a mid-level class have to have a name that matches the source code ('DMDemand' in the example above), the names of subordinate functions are arbitrary so they can be given names appropriate for the concept they represent (eg, 'DeltaNodes', 'DeltaHeight' and 'Stems' in the example above).

The *Plant* and lower level class structure and the use of a configuration file to select and parameterise classes was also used in the original APSIM PLANT template. The additional structuring of classes into mid and low level classes and the development of an IDE for visual configuration is new to PMF.

2.1. Communications and process propagation

Instances of crop models communicate with the APSIM model environment using .NET events which are raised to correspond with management or processing events such as *Sow*, *Process*, *Harvest*, *EndCrop* and *Cut* (Moore et al., 2007). The *Sow* event handler is invoked whenever a crop is to be sown. On this event the crop model constructs and configures its class hierarchy. The bulk of PMF computation is triggered by the *Process* event. In response to this event, the *Plant* class raises a number of its own events (Fig. 2) that propagate processes through the mid-level classes:

1. *DoPhenology* is subscribed to by the *Phenology* class and triggers calculation of daily crop development.
2. *DoPotentialGrowth* is subscribed to by the *Structure* class and each *Organ* classes and triggers the calculation of how much each organs number of constituents, dimensions and mass could change and how much DM and N they may contribute to plant growth.
3. *DoWater* is subscribed to by *Root* and *Leaf* organ classes and triggers calculations of water supply and demand used to determine the extent of water stress.
4. *DoArbitrator* is subscribed to by the *Arbitrator* class and triggers procedures to determine how much DM and N each *Organ* class will actually contribute and receive.
5. *DoActualGrowth* is subscribed to by *Organ* classes, triggering updates of other state variables and moves senesced biomass from *Live* to *Dead* biomass pools.

Propagating processes through the PMF in this way enables flexibility of model structure. If a particular functionality is required, the appropriate class is included and responds to the appropriate events. If that functionality is not required in the crop model the class can be excluded from the configuration and the events that it would have responded to go unheard.

2.2. Phenology

Phenology is the development of crop through a series of phases which exhibit differences in the genesis of organs, nature or environmental response or resource partitioning. Crops differ widely in both the number of phases they contain and the types of behaviour exhibited within these phases. Similarly, model developers differ in their opinion of the number and type of stages that characterise crop phenological development. Phenology may not be required for simple crop models (such as Slurp) that do not change their function over time. When phenology is required in a crop model, it is represented by including the *Phenology* class in the plant configuration file with the necessary *Phase* sub-classes (Fig. 1). The number and type of *Phase* sub-classes (Fig. 2) is unlimited by the source code and is at the complete discretion of the model developer. Phenology works on a daily time-step. Each day (when the *DoPhenology* event is invoked) the *Phenology* class interrogates the current *Phase* class to determine if it has reached its *Target*. When it has, it calculates the proportion of the day that was not used to complete that phase and passes it to the next *Phase* class instance. This is repeated each day until all phases have been completed. Rewind actions can also be included to return the crops development to an earlier phase to simulate the effects of defoliation on perennial crops. This structure enables phenology of varying degrees of complexity to be established (Fig. 1). There are a range of *Phase* sub-classes representing different types of phenology which are described in Table 2 and examples of the parameterisation of phases are given in Section 3.

2.3. Structure

The Structure class represents the crops morphology and is required for models where organ classes need this information. The main properties of the *Structure* class include *PlantPopulation*, *MainStemNumberPerPlant*, *MainStemPrimordiaNumber*, *MainStemNodeNumber*, *MainStemBranchNumber* and *Height*.

2.4. Organs

All organ classes use the same *Biomass* sub-class to keep track of their live and dead biomass status (Fig. 2). The *Biomass* sub-class currently deals with DM and N content of organs. In the future it will also deal with P, K and other nutrients. Each of these components of biomass are separated into three types of pools represented by the properties:

Table 2
Description of phase sub-classes used to construct phenology in the PMF. Icons correspond with phases shown in Fig. 1.

Phase class	Properties	Phase completes when:
Germinating	–	Extractable soil water content (mm^3/mm^3) in top layer exceeds zero.
Emerging	• *ThermalTime* • *LagPeriod* • *ShootRate*	Accumulated thermal time (°Cd) exceeds the sum of *LagPeriod* and the *ShootRate* (mm °Cd^{-1}) multiplied by the sowing depth (mm)
Generic	• *Target* • *ProgressRate* • *StressFactors*	Accumulated daily *ProgressRate* (*ProgressRate* × *StressFactors*) exceeds the *Target*.
LeafAppearance	• *RemainingLeaves*	*MainStemNodeNumber* exceeds *FinalLeafNumber* − *RemainingLeaves*
LeafDeath	• -	All leaf cohorts are fully senesced.
PhaseJump	• *DestinationPhase* • *Event*	The specified *Event* occurs. Current phase is set to *DestinationPhase*.
EndPhase	–	An external event (e.g. EndCrop) finishes the crop. Phenology will stay in this phase until then.

- *StructuralDM* and *N* are essential for the growth of the organ. They remain within the organ once it has been allocated and are passed from *Live* to *Dead* pools as the organ senesces.
- *MetabolicDM* and *N* are essential for growth and their concentration can influence the function of organs (e.g. photosynthetic efficiency of the leaf depends on *MetabolicN* content). *MetabolicDM* and *MetabolicN* may be reallocated (moved to another organ upon senescence of this organ) or retranslocated (moved to another organ at any time when supplies do not meet the structural and metabolic DM demands of growing organs).
- *NonStructuralDM* and *N* are non-essential to the function of an organ. They will be allocated to an organ only when all other organs have received their *Structural* and *Metabolic* allocations and may be reallocated or retranslocated.

A range of organ classes have been developed for the PMF. The simplest is the *GenericOrgan* which contains properties of biomass status and daily biomass demand and supplies (Table 3). The reproductive organ is similar to the generic organ but determines its biomass demands in a different way and also has a *GrainNumber* property. *SimpleLeaf* and *Root* organs deal with biomass demands the same as *GenericOrgan*. However, *SimpleLeaf* includes properties for *DMSupply* (*photosynthesis*), *LeafAreaIndex*, *GreenCover* and *WaterDemand* (Table 3). It is called simple because it is much less dynamic that the regular *Leaf* organ. The *Root* organ has additional properties of *Depth*, *RootLengthDensity*, *NSupply and WaterSupply*. More detail of the different way these organs can be set up is given in Section 3.

A phytomer type *Leaf* class has also been developed for the PMF. It has the same basic properties as *SimpleLeaf* but predicts the area and biomass as the tally of areas and biomass of separate cohorts of leaves. A cohort of leaves is represented by a main-stem node position and branch leaves are kept in the same cohort as the main-stem leaf appearing at the same time (Lawless et al., 2005). The *Leaf* class delegates the status and function of individual cohorts into *LeafCohort* sub-classes (Fig. 2). Further detail of the *Leaf* class is given in Oat model example below.

Table 3
Examples of Oragn classes in the Plant Modelling Framework including some of their key properties and the a brief description of the source code procedure associated with the computation of each property. Icons correspond with organs shown in Fig. 1.

Organ class	Properties	Associated source code procedure
Generic	Biomass	Add N and DM types allocated by the arbitrator and remove senescence calculated from *SenescenceRate*
	StructuralDMDemand	Return *StructuralDMDemand* Function value
	NonStructuralDMDemand	Return product of *StructuralDMFraction* and *StructuralDM* less current *NonStructuralDM*
	StructuralNdemand	Return product of *MinimumNConc* and *StructualDMDemand*
	NonStructuralNDemand	Return product of *MaximumNConc* and current DM less current N
	NReallocationSupply	Return product of *ReallocationFactor* and *NSenescence* calculated for that day.
	NRetranslocationSupply	Return product of *RetranslocationFactor* and current *NonStructuralN*.
Reproductive	GrainNumber	Return GrainNumber function value
	StructuralDMDemand	PotentialDMFillingRate MaximumGrainSize[1]
	StructuralNdemand	PotentialNFillingrate MinimumNConc MaximumNConc[1] GrainNumber
Simple leaf	Biomass properties	As for generic organ
	DMSupply	Return value of *Photosynthesis* function
	LeafAreaIndex	Return *LAIfunction* value
	GreenCover	Return *GreenCover*
	WaterDemand	Return Transpiration demand calculated by micro-metrological component[2]
Root	Biomass properties	As for generic organ
	Depth	Add daily *RootExtnesionRate* value to current depth[3]
	NSupply	Obtains extractable mineral N from each soil layer within root *Depth* [4]
	WaterSupply	Obtains extractable soil water from each soil layer within root *Depth* [5]

References. [1] Asseng et al. (2002). [2] Snow and Huth (2004). [3]Robertson et al. (1993). [4]Probert et al. (1998a). [5]Meinke et al. (1993).

2.5. Biomass partitioning

Organ classes have been designed around an arbitrator interface that provides a core set of properties that *Organ* classes may need to provide to the *Arbitrator* class with their biomass supplies and demands and to receive biomass allocations from the *Arbitrator* (Fig. 4):

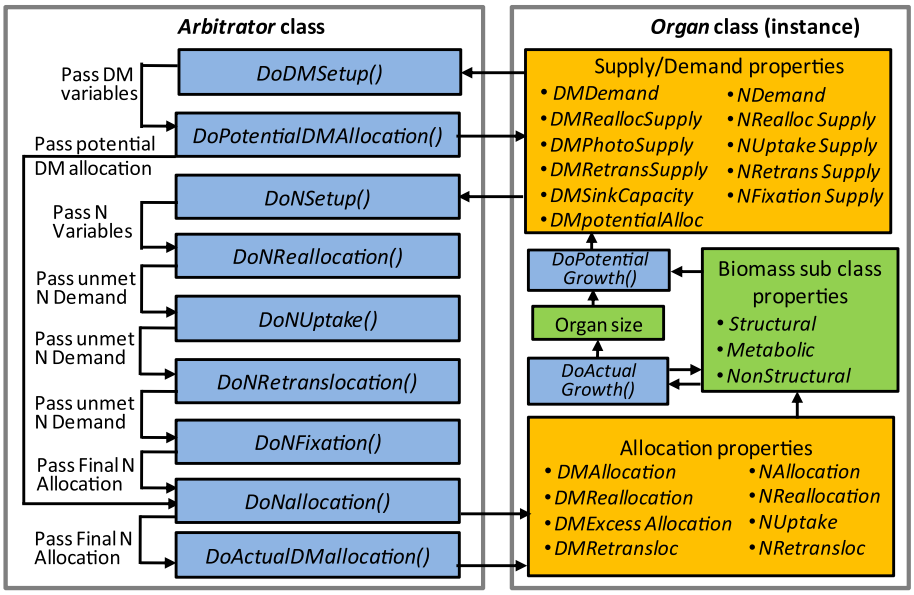

Fig. 4. Schematic showing procedure for biomass partitioning arbitration. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties and blue boxes contain events which are triggered during the daily time step of the model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- Biomass supply and demand properties, which are set by each organ class and interrogated by the *Arbitrator* class:
  - *DMSupply* contains supply types *Photosynthesis*, *Retranslocation* and *Reallocation*
  - *DMDemand* contains types *Structural*, *Metabolic* and *NonStructural*
  - *DMPotentialAllocation* contains types *Structural*, *Metabolic* and *NonStructural* which are used to calculate *NDemand* types
  - *NSupply* contains supply types *Reallocation, Uptake, Fixation and Retranslocation*
  - *NDemand* containing *Structural, Metabolic* and *NonStructural* types.
- Biomass allocation properties that are set by the *Arbitrator* class to deliver biomass allocations to each organ class:
  - *DMAllocation* contains *Allocation, Reallocation* and *Retranslocation* types
  - *NAllocation* contains *Allocation, Reallocation, Uptake, Fixation* and *Retranslocation* types.

To minimise the amount of code needed, these arbitrator interface properties are all contained within a *BaseOrgan* class and all other organ classes inherit from this. Each property has zero values by default and code written into the appropriate properties of inheriting *organ* classes overwrites these defaults. Specific organ classes only contain procedures to assign values to those supplies and demands that are relevant. For instance the *Leaf* class is currently the only one that contains a *PhotosynthesisDMSupply* and the *Root* class is the only one containing an *UptakeNSupply*. However, if models require other organs to photosynthesise or take up N, these can be added into other *Organ* classes without needing to reengineer the arbitrator interface.

The *Arbitrator* regulates the partitioning of biomass among *Organs* using resource supplies and demands from each *Organ* instance (Fig. 4). The growth of a plants DM is the minimum of DM supply from photosynthesis and DM demands from structural growth and the capacity to store non-structural DM (Gent and Seginer, 2012). The PMF arbitrator extends this idea to the organ scale where the DM supply becomes that of photosynthesis plus the reallocation and remobilisation of non-structural DM from other organs. In situations where DM demands exceed supplies the *Arbitrator* class can partition to organs based on their demand relative to other organs or on a user-specified priority ranking. The partitioning of N is inherently linked to DM partitioning in the PMF *Arbitrator* class based on the partitioning concepts of the SIRIUS modelling approach (Jamieson et al., 1998).

The *Plant* class first invokes *DoPotentialGrowth* (Fig. 2) when each organ determines how much its biomass, dimensions and number properties could increase and sets the values of *DMSupply* and *Demand* properties (Fig. 4). Next it calls *DoDMArbitration* and *DoNArbitration* which trigger a sequence of functions in the *Arbitrator* class to determine how much DM and N will be allocated to each organ (Fig. 4). *DoPotentialDMAllocation* takes *PhotosynthesisDMSupply* and *ReallocationDMSupply*, partitions these to organs based on their *Structural* and *MetabolicDMDemands* and partitions any *SurplusDMSupply* to organs based on their *NonStructuralDM* demand. If some *DMSupply* is still unallocated this remains unallocated with the assumption that the plant would down regulate photosynthesis due to lack of sink capacity in such cases (Gent and Seginer, 2012). If the *Structural* and *MetabolicDMDemand* are not met *RetranslocationDMSupply* is used to meet these demands. Then *DoNReallocation*, *DoNUptake*, *DoNRetranslocation*, and *DoNFixation* determine organ *NAllocation* from each of these supplies. *DoActualDMAllocation* takes *NAllocations* for each organ, determines if this is enough to maintain *MinNConc* and if not the *DMAllocation* is constrained and *SurplusDM* discarded. This assumes that under severe N stress photosynthesis would be down regulated due to N inadequacy limiting sink strength (Gent and Seginer, 2012; Jamieson et al., 1998).

Currently arbitration has only been implemented to deal with N supply. However, P, K and other nutrients can also be included in the same way with each organ registering supplies and demands, the arbitrator allocating these and constraining DM allocations to maintain minimum nutrient concentrations in organs.

2.6. Water stress responses

Water stress is represented in PMF using a *WaterSupplyDemandRatio* property calculated as supply/demand (Brown et al., 2009; Wang et al., 2004). The supply comes from the *Root* organ (Table 3) and demand from the MicroMet model in APSIM (Snow and Huth, 2004). Values ≥ 1 mean water supply is able to met demand and values <1 infer some degree of stress. Processes that are influenced by water stress have their rates multiplied by stress factors. The PMF allows water stress factors to be included to influence any process. Currently they are applied to photosynthesis, leaf area expansion, stem extension, leaf senescence, branch mortality and N fixation. These responses to water stress are configured in the crop.xml files (e.g. the 'FW' function under 'Photosynthesis' in the 'Leaf' organ of the lucerne model, Fig. 1).

3. Crop model examples

3.1. Slurp

The simplest crop model implemented in PMF is Slurp (named after the action of slurping water from the soil) which consists of *SimpleLeaf* and *Root* organs. This model is for the purpose of doing water balance studies where detailed representation of plant processes are not required, e.g. Snow et al. (2007). The functionality given by *Arbitrator*, *Phenology* and *Structure* classes is not needed so these classes are omitted from the Slurp configuration. The values of *LeafAreaIndex*, *Height* and *RootDepth* are given to *Leaf* and *Root* organs by *Constant* functions.

3.2. Oat

The oat model is an example of a complex crop model, currently under development, to demonstrate the functionality of the cohort leaf class.

3.2.1. Phenology

The structure of the phenology model is displayed in Fig. 1. The 'Germinating' phase is represented by a *Germination* phase class and the 'Emerging' phase by an *Emerging* phase class (Table 2). The emergence phase leads into a 'Vegetative' phase which is currently represented by a *Generic* phase class (Table 2) with a constant 70 °C *ThermalTimeTarget*. This can be expanded with more detailed functionality in the future to represent vernalization responses of sensitive cultivars. Next is the 'EarlyReproductive' phase, which is also a *GenericPhase* class with a *ThermaltimeTarget* that uses a *LinearInterpolation* function to decrease from 650 °Cd to 400 °Cd, as photoperiod increases from 10 to 16 h. This captures the photoperiod sensitivity that is displayed by oats (Martin et al., 1998a). *FinalMainStemNodeNumber* is set in the *Structure* class on the completion of the early reproductive phase (Section 2.3) and the following 'PseudoStemExtension' phase is a *LeafAppearance* phase class (Table 2) that finishes when flag leaf has appeared. Following this there are a series of *Generic* phase sub-classes with *Constants* for *ThermaltimeTarget* representing grain development and ripening phases (Fig. 1).

3.2.2. Structure

In the oat model *MainStemPrimordiaNumber* is assumed to represent the number of primordia committed to becoming leaves. In cereals the commitment of primordia to becoming leaves increases as a linear function of leaf appearance until floral initiation, when the fate of all primordia is set (Brown et al., 2013; Jamieson et al., 2007). Sonego et al. (2000) showed that final main-stem node number in 'Drummond' oats is related to the modified Haun stage at which floral initiation occurs (FI^{HS}) by:

$$FinalMainStemNodeNumber = 2.8 + 1.2 \times FI^{HS}$$

To model this, *MainStemPrimordiaNumber* is set to 3 at the time of emergence and *MainStemPrimordiaInitiationRate* is calculated by dividing *MainStemNodeAppearanceRate* by 1.2 ('PrimordiaPerMainstemNode' function in Fig. 1). *MainStemPrimordiaNumber* is fixed at floral initiation (the end of the 'EarlyReproductive' phase) which determines the upper limit for *MainStemNodeNumber* (Fig. 5a). Thus, the photoperiod response that is programmed into the early reproductive phase directly affects *FinalMainStemNodeNumber* and the timing of flag leaf and subsequent anthesis (Section 3.2.1). The value of *MainStemNodeAppearanceRate* for predicting *MainStemNodeNumber* (Fig. 5a) is delivered by a *Multiply* function ('MainStemNodeAppearanceRate' in Fig. 1) that provides the product of 'BasePhyllochron' (*Constant* function) and 'LeafStageFactor' (*LinearInterpolation* function) that decreases as *MainStemNodeNumber* increases (Jamieson et al., 1995; Martin et al., 1998b).

The 'BranchingRate' (number of new branches produced each day, Fig. 1) is given by a *Multiply* function that returns the product of a 'PotentialBranchingRate', a 'ShadingFactor' and a 'WaterStressFactor'. In oats, the first tiller appears when the 4th main-stem node appears and in wide spaced plants a further tiller may appear with each main-stem node until floral initiation. To reproduce this process the PotentialBranchingRate is a *PhaseBasedLookup* function with a *LinearInterpolation* function returning a value (increasing from 0 for the first 3 main-stem nodes, to 1 for all subsequent nodes) from emergence until floral initiation and returning a *Constant* of zero for subsequent phases. As plant spacing decreases the extent of tillering on individual plants decreases also. To capture this, the 'ShadingFactor' is a *LinearInterpolation* function which has a value of 1.0 when *GreenCover* is between 0 and 0.5, decreasing to 0 at a *GreenCover* of 0.8. This means, at lower populations, plants will produce more tillers than at higher populations. 'ShadeInducedBranchMortality' and 'DroughtInducedBranchMortality' functions are also included (Fig. 1) with *LinerInterpolation* functions returning positive values at high *GreenCover* and low *WaterSupplyDemandRatio* respectively.

3.2.3. Leaf organ

The Oat model uses the Leaf organ class containing the phytomer leaf model. The duration of expansion of oat leaves is twice that of the phyllochron (Sonego et al., 2000) so the *GrowthDuration* value is delivered to each *Cohort* class by a *Multiply* function returning the product of 'Phyllochron' and a *Constant* of 2.0. This means the difference between number of tips and number of fully expanded leaves will also be 2 (Fig. 5a). The *MaximumSize* of each *Cohort* is currently set as a function of node position but will be changed to a function of developmental stage to capture the effect of changes in final leaf number on the position of the largest leaf. The *LagDuration*, when the cohort area remains at its maximum is a *Constant* 800 °Cd and the *SenescenceDuration* a *Constant* 600 °Cd. The 'ShadeInducedSenescenceRate' uses a *LinerInterpolation* function to return a value of 0 when *OverlyingCover* is < 0.8, increasing to a rate of 0.02 for covers from 0.8 to 0.97 and increasing to 0.1

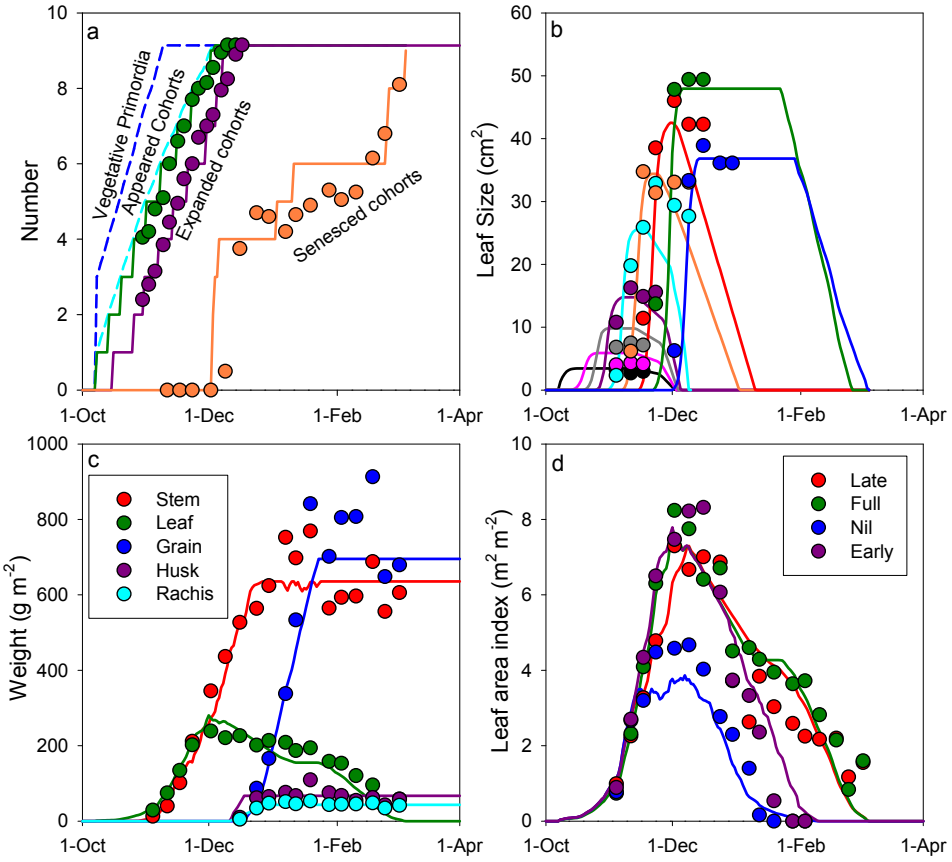

Fig. 5. Oat model predictions compared with observations for a) the initiation of primordia (when cohorts are initialised), the appearance of that cohort and the completion of its expansion and senescence; b) the expansion and senescence of leaves at subsequent main stem positions; c) the growth of organs; d) the leaf area index of treatments receiving early, late, full or nil irrigation treatments. Observed data taken from trial described by Martin et al. (2001).

when *OverlyingCover* exceeds 0.97. Thus, when a crop has a high stem population and many leaves, the lower leaves will senesce away keeping *LeafAreaIndex* at realistic values. A comparison of the size of each leaf position relative to that predicted for each cohort is shown in Fig. 5b and the net *LeafAreaIndex* predicted from this model for different drought treatments shown in Fig. 5d.

3.2.4. Other organs

The 'Grain' in the oat model (Fig. 1) is represented by a *Reproductive* organ class (Table 3). The *GrainNumber* given by a *Multiply* function which returns the product of *Structure.TotalStemPopulation* (using a *VariableReference* function), the number of 'PaniclesPerStem' (a *Constant* of 0.83), the number of 'SpikeletsPerPanicle' (a *Constant* of 25) and the number of 'GrainsPerSpikelet' (a *LinearInterpolation* returning a value which increases from 0 at flowering up to 1.95, 250 °Cd after flowering).

The 'Stem' is represented by the *GenericOrgan* class (Table 3). Stem weight is correlated with node number (main-stem and branched nodes) over a range of water stress conditions giving differences in internode length (Martin, unpublished data). This shows there is plasticity in the specific length (mm g^{-1}) of nodes so when stress reduces node expansion the nodes can become denser. However, under severe stress the correlation between node number and stem mass changes suggesting there is a limitation to how dense internodes can become. To capture this, the *StructuralDMDemand* is determined using a *Minimum* function which returns the lowest of 'DeltaNodeNumber' and 'DeltaHeight' limited demands (Fig. 1). The 'Husk', 'Rachis' and 'Peduncle' organs are also represented by the *GenericOrgan* class but with simple representations of *StructuralDMDemand* using a function class called *PopulationBasedDemand*. This function has parameters of *MaximumOrganSize*, *StartStage* and *GrowthDuration*. Linear growth is assumed so daily *StructuralDMDemand* of each organ is calculated as the product of *PotentialDailyGrowth* (*MaximumOrganSize/GrowthDuration*), *ThermalTime*, a *WaterStressFactor* and the *TotalStemPopulation* (assuming each stem has one of these organs). An example of the biomass accumulation patterns that result from these organ class parameterisations is shown in Fig. 5c.

3.3. Lucerne

Lucerne is a perennial crop that is frequently cut and the extent of biomass partitioning to below ground organs varies among the consecutive regrowth periods throughout the year (Teixeira et al., 2007a). It is also a crop for which recent advances in physiology had been included into another model (Teixeira et al., 2009). The lucerne model is used to demonstrate how feasible it is to reproduce the functionality of an alternative model of intermediary complexity into PMF.

3.3.1. Phenology

The *Germination, Emerging,* and *Generic Phase* classes (Table 2) are used to construct the development of the lucerne crop from sowing – harvest ripe (Fig. 1) with phases parameterised as described by Teixeira et al. (2011). Lucerne differs from the other crop models described because it is perennial (as opposed to annual) and regrowth must be modelled. To achieve this there are a set of regrowth phases that the crop proceeds into once grain

production is complete (Fig. 1). The crop will also reset phenology to the 'RegrowthVegetative' phase each time a defoliation event is specified by the APSIM Manager module. When regrowth reaches the end of the 'RegrowthEarlyReproductive' phase, phenology is then re-set to the 'PodDevelopment' phase (Fig. 1) so the same phases can be used to represent the seedling and regrowth crop.

3.3.2. Leaf organ

In the case of lucerne, the complexity of seasonal changes of leaf size and branching dynamics makes it challenging to parameterise the phytomer model in the *Leaf* class. Instead the *SimpleLeaf* class (Table 1) was used with functions included to provide changes in *LeafAreaIndex, DMSupply* and *DMDemand* (Fig. 1). Teixeira et al. (2007b) showed the daily increase in *LeafAreaIndex* of lucerne is closely related to thermal time in unstressed crops with *MainStemPopulation* greater than 800 stems/m^2. To capture this *LeafAreaIndex* is modelled using an *Accumulate* function ('LAI' in Fig. 1) which keeps a tally of daily increments ('DeltaLAI') in response to *ThermalTime*. 'DeltaLAI' uses a *Multiply* function to return the product of (i) potential leaf area expansion rate ('LAER', a *Constant* value of 0.016 m^2 leaf per m^2 of soil per °Cd), (ii) a 'PhotoPeriodAdjFact' (*LinearInterpolation* retuning a value of 1.0 for photoperiods > 12, reducing to 0 at a photoperiod of 10.0 h), (iii) a 'LAIAdjFact' (*LinearInterpolation* factor to reduce expansion rates at LAI < 1) and (iv) a 'SeedlingAadjFact' which was a *PhaseLookup* class returning a value of 0.6 for seedling phases and 1.0 for regrowth phases. The *Accumulate* class subtracts a specified proportion from value when a defoliation event is sent from the APSIM model so LAI is decreased in response to such events. The LAI model is relatively simple collection of functions but was able to reproduce the LAI dynamics of a lucerne crop for 2 years from sowing (Fig. 6a).

The 'Leaf' was as *SimpleLeaf* organ given a *DMSupply* by including the *RUEModel* function ('Photosynthesis' in Fig. 1) parameterised using the RUE responses described by Brown et al. (2006). The 'DMDemand' for leaf was provided by including a *PartitionFractionDemand* function which returns a demand that is the product of *DMSupply* and a *PartitioningCoefficient* which was adapted from the one described by Teixeira et al. (2009).

3.3.3. Stem and root organs

The 'Stem' organ used the *GenericOrgan* class (Table 3) with a *PartitionFractionDemand* function to set its 'DMDemand' as described by Teixeira et al. (2009). The tap and fine roots were represented with the *Root* organ class which provides mineral N and water supplies to the arbitrator and keeps track of below ground biomass. Lucerne shows a seasonal fluctuation in below ground DM with increases in the autumn and reductions in winter and spring (Fig. 6c). This was modelled by using two collections of functions to reproduce seasonal patterns of (i) biomass partitioning to roots and (ii) root senescence plus maintenance respiration adapted from Teixeira et al. (2009). This set up was able to reproduce the complex seasonal patterns of shoot and root DM observed in a lucerne crop (Fig. 6) without requiring any changes to the source code of *Root* organ class.

3.3.4. Flexibility for further model development

This example shows that the PMF framework enabled flexibility to successfully implement a previously developed lucerne model (Teixeira et al., 2009) for crops growing under unconstrained water, nutrients and harvest management conditions. This basic set up enables flexible expansion of the model capability. Specifically, to account for sub-optimal growth conditions, lucerne reserve organs (e.g. crowns and taproots) have to be implemented to store nitrogen, a root nodule organ is required to fix atmospheric nitrogen and water stress responses have to be parameterised. This will enable the model to deal with the effects of severe grazing on root biomass and shoot regrowth. In addition, the uncoupling of root respiration and senescence, the effects of low population on canopy growth, and inclusion of varieties with contrasting dormancies, as described in the existing version of APSIM, can be easily transferred to this pilot lucerne model.

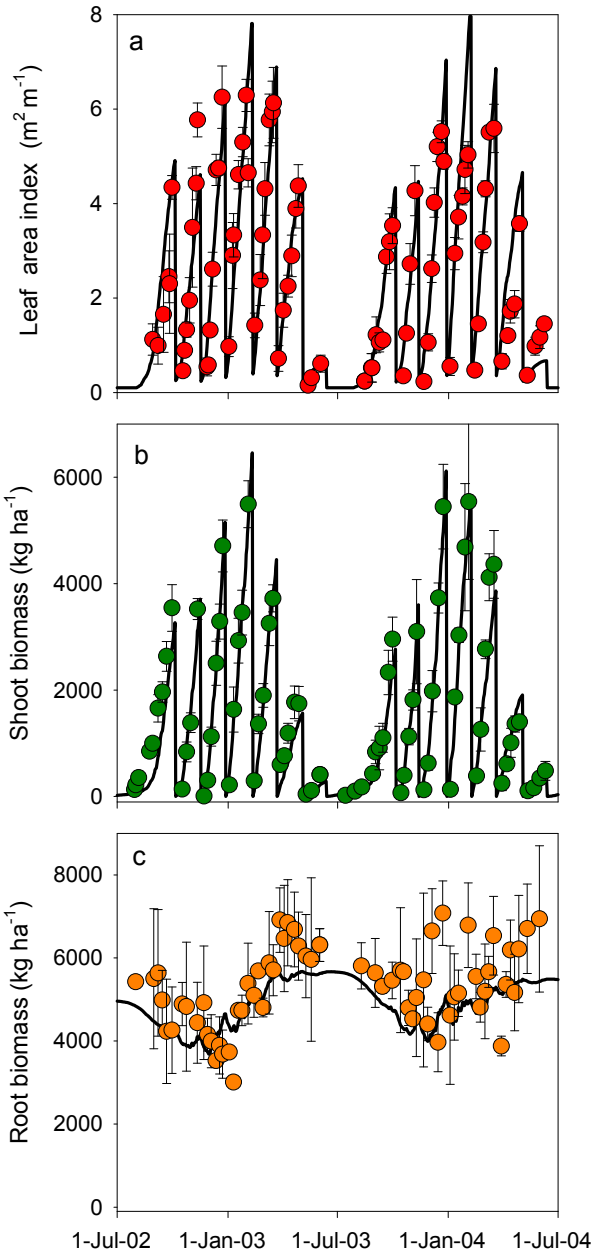

Fig. 6. Observations and predicted values of a) leaf area index; b) shoot biomass; and c) root biomass of lucerne. Observed data from Teixeira et al. (2007a) and Teixeira et al. (2007b).

3.4. Wheat model

Wheat was initially a series of translations of the CERES wheat model (Ritchie and Otter, 1985) into the Fortran77-based APSIM framework (Asseng et al., 1998; Keating et al., 2001; Meinke et al., 1997). Upon the development of the GCROP template (Wang et al., 2003), the science of these was translated into this generic

crop template which also represented a move to Fortran90. This was then merged into a code base which also allowed for simulation of legumes and perennial plants (Robertson et al., 2002) to make an even more generic modelling framework and represented another language move into C and then C++. This model evolution has been driven by the advantages of moving to more generic designs, including object oriented and pattern-based development at each step. Science developments have been implemented into the model over this time also with most functionality being maintained from one version to the next. Ongoing testing (Holzworth et al., 2014, 2011) has ensured that model performance and integrity has been maintained during each evolutionary step. The documentation of APSIM wheat is available online (www.apsim.info).

The decision was made to further evolve the wheat model into the PMF to make use of its better design. However, the wheat model was already extensively validated and has a large user base, and so it was important to ensure that model performance was maintained. Whilst it is desirable to implement the wheat model using the PMF classes described above, doing this and re-testing the model would require a considerable effort. Processes are being developed to help accelerate these processes. However, until these become available, it was decided to conduct a software port of the existing wheat model into PMF by using PMF components wherever possible and porting other required processes into C#. It was possible to use the PMF phenology classes to represent development and capture much of the remaining functionality in function classes. The result is a set of new mid-level classes (Fig. 2) that replicate the existing wheat model.

The PMF Wheat model produces outputs that are identical to those of the existing wheat model in the standard APSIM wheat model validation set containing 164 simulations for a wide range of environments and treatments (Supplementary material). The predicted yields for above ground biomass (Fig. 7a), grain yield (Fig. 7b), and nitrogen content in above ground biomass (Fig. 7c) and grain (Fig. 7d) all showed good agreement with observed values. These results show that this intermediate step in the model's evolution can still be used with confidence by the APSIM user community whilst we continue to move toward using more of the standard PMF classes.

4. Discussion

Plant models are undergoing continual evolution as progress is made in the science that they represent and the software that is used to implement them. The notion of a generic crop template has evolved as a means of improving the efficiency of development of models and maintenance of source code. However, some crop templates such as CROPGRO (Boote et al., 1998) GCROP (Wang et al., 2003) and PLANT (Robertson et al., 2002) have been overly prescriptive of the way models were structured and the data that is needed to represent them. Generic crop templates have also suffered from developers and maintainers not wanting to make changes to the source code to overcome a problem or expand the science for one crop for fear of affecting the performance of other crop models built using the same code base. The PMF is the next generation of crop template that attempts to achieve the same

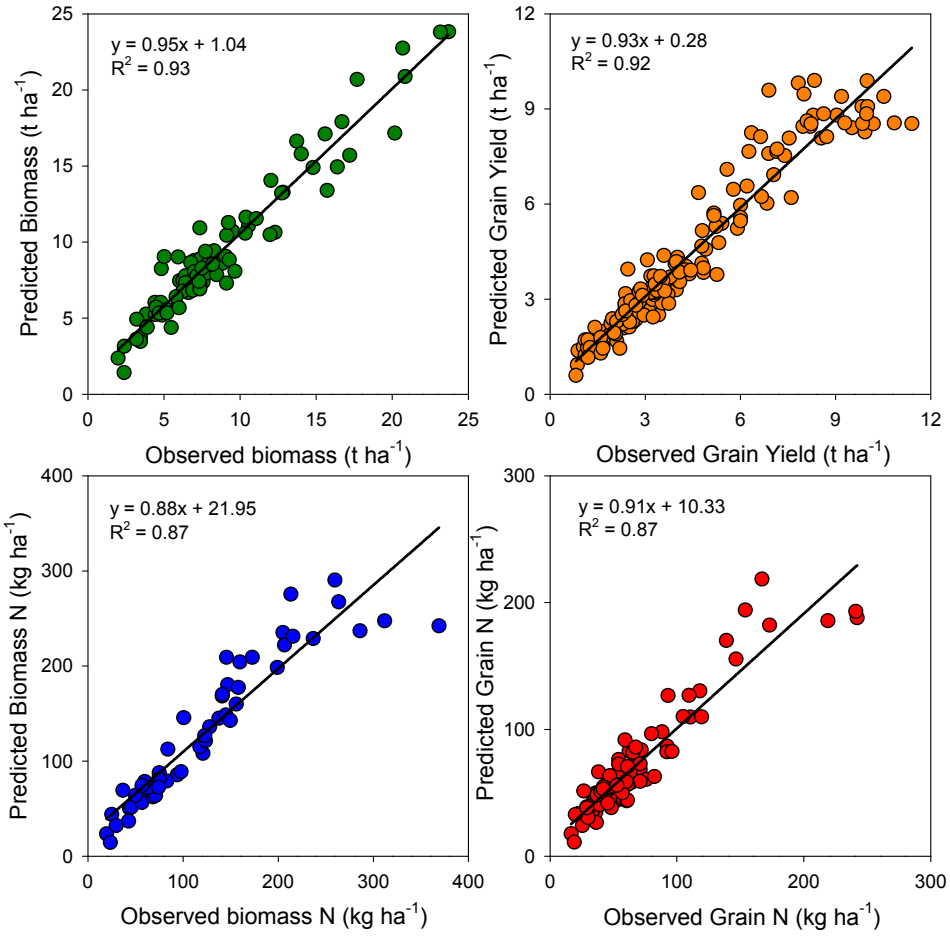

Fig. 7. Wheat model predictions plotted against observed biomass, grain yield, biomass nitrogen and grain nitrogen.

objectives as previous crop templates while providing more flexibility in the way models are structured and reducing the need for model developers to write or compile code. A number of design goals were outlined in the introduction and in the sections below we consider how successful the PMF has been in achieving these:

4.1. Enable models to be established at different levels of complexity

Creating models with different levels of complexity was considered important so software developers were not limiting the approaches that crop physiologists and modellers could take in the development of crop models. This flexibility is achieved in three ways. Firstly by using a fixed interface for the communication between top and mid level organs (Fig. 1) without any mandatory mid-level classes allows functionality to be added or subtracted as needed. Secondly by providing a range of mid-level classes with different sets of functionality at different levels of detail (eg *SimpleLeaf* vs phytomer *Leaf* classes) or with different approaches (e.g the Wheat model classes) allows the developer to choose an appropriate level of complexity. Thirdly, by programming the mid level classes generically, the developer can choose a combination of *Function* classes that matches the required level of complexity. (Section 2). The examples given in this paper show models ranging from very simple (slurp), intermediate (lucerne) and detailed (oat and wheat) and all have been successfully implemented in PMF showing this design goal has been achieved. This flexibility is not unique to PMF with the APES model (Donatelli et al., 2010) also allowing models to be assembled at different levels of complexity. As the types and scale of applications for crop models broadens, modellers will require increasing flexibility in how models are structured and parameterised and it seems fair to expect that software developers must provide tools to enable this flexibility.

4.2. Enable code reuse and minimise the amount of code to maintain

This is a key goal of all generic crop templates (Adam et al., 2012; Jones et al., 2001; Penning de Vries et al., 1989). The PMF classes represent generic elements of the plant structure or function and can be parameterised to represent different sets of functionality for different crops. For example, the oat model shown in Fig. 1 contains four instances of the *GenericOrgan* class (Husk, Rachis, Peduncle and Internodes). In each case, this class is parameterised differently to represent the different kinds of organs. The use of sub-classes to represent repeated data structures and processes as well as the extensive use of function classes also achieves code reuse each time a particular mathematical or procedural computation is made. At this stage we can conclude that PFM has achieved its goal of maximising code reuse. However, as the number of developers using PFM increases it is likely that overlapping functionality will be written because those new to the system will not be fully aware that the current library of functions contains the mechanisms needed. This is a case for good documentation and easy visualisation of the available source code classes. Reflection tags allow for the inclusion of documentation within source code and the rendering of it using auto documentation systems which have been used to produce documentation of some PMF models. However there is also a need for a system to extract comments from the source code and associate them with classes in the IDE so instructions displayed there remain in sync with source code development.

4.3. Externalise the structure and parameterisation of a crop model

Externalisation of model structure and parameterisation (Holzworth and Huth, 2009a) eliminates the need for model developers to compile source code. The PMF is facilitated by generalising the classes and externalising the calculation of the values they use into function classes (Fig. 1). Thus, nearly all of models functionality is determined by which mid-level classes are included and how function classes are combined to provide their values. There are still large parts of the models functionality that are inherent to the structure of the source code within classes and their interfaces (such as the arbitration procedures) and cannot be changed without traditional coding and compiling. However, the PMF enables considerably more flexibility to change models from the configuration files than previous crop templates allowing non-programming model developers more flexibility than was previously available. With the large number of functions and classes available, it can be daunting for new developers to achieve an understanding of which parts to 'click' together when creating a new plant model. This is another case for developing good documentation and tools to ease the developer through this process.

4.4. Provide a framework that enables the easy inclusion of new organ, process and function class alternatives

The PMF contains a set of generic classes that have already been used to build a number of crop models. However, if a modeller wants to change some of the fundamental processes or properties of a class, alternative mid-level or function classes can be written and included in the source code (e.g. the Wheat model, Section 3.4). If developers wish to aggregate processes in a different way, they can write alternative classes that interface with the plant class and use any of the function classes to provide them with values (e.g. the wheat model). This design conflicts with the goal of maximising code reuse because it allows for multiple ways of doing things. However, the need to enable alternative approaches is necessary to allow scientific exploration of modelling approaches and to make development in the PMF environment comfortable for a broad range of developers. The successful achievement of this design goal might contribute to enlargement of the source code and the maintenance burden for the software team. However, the low level function concept will help slow code enlargement and the flexibility that this provides to model developers is seen as a worthwhile benefit.

4.5. Potential problems and misuse of PMF

One possible problem is that making the model structure accessible to more developers is loss of control of the model code with the prospect of developers making unjustified changes to the model then representing it as the released version. The greater ease in setting up models could also encourage ill-informed model development. To partially address this issue we need to make it clear to would-be-developers that a detailed understanding crop physiology is an essential prerequisite to construction of a robust model in PMF (as it is for any model platform). We will also emphasise that best practice should always be followed in the use of models. Specifically, any changes that have been made to a released version of a model must always be clearly detailed and justified. Ultimately though, model performance will be judged by its validation against observed data. Like all major changes to models in APSIM, new or altered PMF models will be reviewed by the APSIM reference panel for scientific merit, design and implementation, before being included in official releases (Holzworth et al., 2014). This helps to minimise any potential misuses of the PMF.

Enabling flexibility by delegating functionality into nests of functions can make it difficult to follow the sequence of program logic leading to difficulties in isolating problems in a model setup. A

general principal of PMF development is to write descriptive error messages whenever a fatal error or fundamental violation is isolated in debugging. The use of .NET also means that any variable can be reported, which facilitates the isolation of unexpected behaviour of a model component. While the IDE approach to developing models makes them accessible to non-programming developers, the XML representation of a plant model is itself a programming language that developers need to learn. For computer programmers familiar with other languages, this is sometimes viewed as limiting and inefficient and may discourage programmers from using PMF.

The delegation of functionality into function configurations moves the code base from the compiled classes to the XML files which means this code is not shared between models. Science ideas that are represented by a nest of functions can be copied from one model to another to share the science but this creates duplication of XML fragments in the plant configuration files potentially leading to fixes needing to be repeated across multiple sets of files. To alleviate this, when a particular pattern of functions occurs in several configuration files, a specialised function is created representing the algorithm and each configuration file is changed to use the new function.

One problem that has hindered past crop templates is the inherent desire to minimise changing code once it provides the basis for validated models. This limits the scope for allowing bug fixes or code improvements to aid other models that use the same code base. The delegation of model structure into plant configuration files solves this problem to a certain extent but there is still structure in the source code. To ensure the code base of PMF was not dictated by the first few models produced, its development to date has not focused on completing and validating crop models. Instead the focus has been on the establishment of a wide range of models to test the generic applicability of the code base, work out any bugs and develop software methods that will enable changes to one model without adversely affecting the performance of another. As such, a number of models have been established in PMF including Oat, Lucerne, Potato (Brown et al., 2011), Field Pea, Kale, Barley, Grape, E.Grandis, Chickpea, Broccoli (Huth et al., 2009), French Bean, Wheat and OilPlam (Huth et al., in this issue).

4.6. Future development

Now that the code base is reaching a point of stability, the focus will move onto the completion of some of these models and the migration of existing APSIM crop models into the PMF. To achieve this some work is still required to reconcile the alternative approaches taken in the arbitration of DM and N allocation in PLANT and the PMF. The process of migration could also be accelerated by including a *LeafAreaIndex* function for the *SimpleLeaf* organ (Table 3) that is analogous to that used in the PLANT template. This will mean migration will not need to involve the parameterisation of a completely new canopy model (Table 3).

The PMF has been moved into the next generation of APSIM (Holzworth et al., 2014) and will form the basis of many new and upgraded models of the plant based components of the APSIM model. This new generation of APSIM will offer an improved model validation and testing procedure. In combination with the ease of model development in the PMF, this will enable faster progress improving the science content and reliability of APSIM crop models. Functionality that will be added to the PMF over the next few years includes the ability to simulate multi species crops. To do this a separate component is being developed to arbitrate inter-plant resource allocation. It will interface with multiple instances of PMF models and provide them with their portion of the daily radiation, water and nitrogen they can obtain for their daily processes. Another feature to be added is predicting responses to P and K nutrition. This has not been included yet but the organ/arbitrator interface and *Arbitrator* class have been designed so these processes can be included in the future.

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.envsoft.2014.09.005

References

Adam, M., Corbeels, M., Leffelaar, P.A., Van Keulen, H., Wery, J., Ewert, F., 2012. Building crop models within different crop modelling frameworks. Agric. Syst. 113, 57–63.

Adam, M., Ewert, F., Leffelaar, P.A., Corbeels, M., van Keulen, H., Wery, J., 2010. CROSPAL, software that uses agronomic expert knowledge to assist modules selection for crop growth simulation. Environ. Model. Softw. 25 (8), 946–955.

Asseng, S., Bar Tal, A., Bowden, J.W., Keating, B.A., van Herwaarden, A., Palta, J.A., Huth, N.I., Probert, M.E., 2002. Simulation of grain protein content with APSIM-Nwheat. Eur. J. Agron. 16 (1), 25–42.

Asseng, S., Keating, B.A., Fillery, I.R.P., Gregory, P.J., Bowden, J.W., Turner, N.C., Palta, J.A., Arbrecht, D.G., 1998. Performance of the APSIM-wheat model in Western Australia. Field Crops Res. 57, 163–179.

Boote, K.J., Jones, J.W., Hoogenboom, G., 1998. Simulation of crop growth: CROPGRO model. In: Peart, R.M., Curry, R.B. (Eds.), Agricultural Systems Modelling and Simulation. Marcel Dekker, Inc, New York, U.S.A, pp. 651–692.

Brisson, N., Gary, C., Justes, E., Roche, R., Mary, B., Ripoche, D., Zimmer, D., Sierra, J., Bertuzzi, P., Burger, P., Bussiere, F., Cabidoche, Y.M., Cellier, P., Debaeke, P., Gaudillere, J.P., Henault, C., Maraux, F., Seguin, B., Sinoquet, H., 2003. An overview of the crop model STICS. Eur. J. Agron. 18, 309–332.

Brown, H., Jamieson, P., Brooking, I., Moot, D., Huth, N., 2013. Integration of molecular and physiological models to explain time of anthesis in wheat. Ann. Bot. 112, 1683–1703.

Brown, H.E., Huth, N., Holzworth, D., 2011. A potato model build using the APSIM Plant.NET framework. In: Chan, F., Marinova, D., Anderssen, R.S. (Eds.), MODSIM2011, 19th International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, Perth, pp. 961–967.

Brown, H.E., Moot, D.J., Fletcher, A.L., Jamieson, P.D., 2009. A framework for quantifying water extraction and water stress responses of perennial lucerne. Crop Pasture Sci. 60 (8), 785–794.

Brown, H.E., Moot, D.J., Teixeira, E.I., 2006. Radiation use efficiency and biomass partitioning of lucerne (*Medicago sativa*) in a temperatre climate. Eur. J. Agron. 25 (4), 319–327.

David, O., Ascough II, J.C., Lloyd, W., Green, T.R., Rojas, K.W., Leavesley, G.H., Ahuja, L.R., 2013. A software engineering perspective on environmental modeling framework design: the Object Modeling System. Environ. Model. Softw. 39, 201–213.

Dolling, P.J., Robertson, M.J., Asseng, S., Ward, P.R., Latta, R.A., 2005. Simulating lucerne growth and water use on diverse soil types in a Mediterranean-type environment. Aust. J. Agric. Res. 56 (5), 503–515.

Donatelli, M., Russell, G., Rizzoli, A.E., Acutis, M., Adam, M., Athanasiadis, I.N., Balderacchi, M., Bechini, L., Belhouchette, H., Bellocchi, G., Bergez, J.E., Botta, M., Braudeau, E., Bregaglio, S., Carlini, L., Casellas, E., Celette, F., Ceotto, E., Charron-Moirez, M.H., Confalonieri, R., Corbeels, M., Criscuolo, L., Cruz, P., diGuardo, A., Ditto, D., Dupraz, C., Duru, M., Fiorani, D., Gentile, A., Ewert, F., Gary, C., Habyarimana, E., Jouany, C., Kansou, K., Knapen, R., LanzaFilippi, G., Leffelaar, P.A., Manici, L., Martin, G., Martin, P., Meuter, E., Mugueta, N., Mulia, R., VanNoordwijk, M., Oomen, R., Rosenmund, A., Rossi, V., Salinari, F., Serrano, A., Sorce, A., Vincent, G., Theau, J.P., Thérond, O., Trevisan, M., Trevisiol, P., Evert, F.K., Wallach, D., Wery, J., Zerourou, A., 2010. A Component-based Framework for Simulating Agricultural Production and Externalities. Springer, Dordrecht, The Netherlands.

Gent, M.P., Seginer, I., 2012. A carbohydrate supply and demand model of vegetative growth: response to temperature and light. Plant Cell Environ. 35, 1274–1286.

Holzworth, D., Huth, N., 2009a. Reflection + XML Simplifies development of the APSIM generic PLANT Model. In: Anderssen, R.S., Braddock, R.D., Newham, L.T.H. (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: Cairns, Australia, pp. 887–893.

Holzworth, D., Huth, N., deVoil, P., Zurcher, E., Herrmann, N., McLean, G., Chenu, K., van Oosterom, E., Snow, V., Murphy, C., Moore, A., Brown, H., Whish, J., Verrall, S., Fainges, J., Bell, L., Peake, A., Poulton, P., Hochman, Z., Thorburn, P., Gaydon, D., Dalgliesh, N., Rodriguez, D., Cox, H., Chapman, S., Doherty, A., Teixeira, E., Sharp, J., Cichota, R., Vogeler, I., Li, F., Wang, E., Hammer, G., Robertson, M., Dimes, J., Carberry, P., Hargreaves, J., MacLeod, N.C.M., Harsdorf, J., Wedgewood, S., Keating, B., 2014. APSIM – evolution towards a new generation of agriculutral systems simulation. Environ. Model. Softw. 62, 327–350.

Holzworth, D., Huth, N.I., 2009b. Reflection + XML Simplifies Development of the APSIM Generic PLANT Model. 18th World IMACS/MODSIM Congress, Cairns, Australia.
Holzworth, D.P., Huth, N.I., de Voil, P.G., 2010. Simplifying environmental model reuse. Environ. Model. Softw. 25 (2), 269–275.
Holzworth, D.P., Huth, N.I., deVoil, P.G., 2011. Simple software processes and tests improve the reliability and usefulness of a model. Environ. Model. Softw. 26 (4), 510–516.
Huth, N.I., Henderson, C., Peake, A., 2009. Development And Testing of a Horticultural Crop model Within APSIM. 18th world IMACS/MODSIM Congress, Cairns, Australia.
Jamieson, P.D., Brooking, I.R., Porter, J.R., Wilson, D.R., 1995. Prediction of leaf appearance in wheat: a question of temperature. Field Crops Res. 41 (1), 35–44.
Jamieson, P.D., Brooking, I.R., Semenov, M.A., McMaster, G.S., White, J.W., Porter, J.R., 2007. Reconciling alternative models of phenological development in winter wheat. Field Crops Res. 103 (1), 36–41.
Jamieson, P.D., Semenov, M.A., Brooking, I.R., Francis, G.S., 1998. Sirius: a mechanistic model of wheat response to environmental variation. Eur. J. Agron. 8, 161–179.
Jones, J.W., Hoogenboom, G., Porter, C.H., Boote, K.J., Batchelor, W.D., Hunt, L.A., Wilkens, P.W., Singh, U., Gijsman, A.J., Ritchie, J.T., 2003. The DSSAT cropping system model. Eur. J. Agron. 18, 235–265.
Jones, J.W., Keating, B.A., Porter, C.H., 2001. Approaches to modular model development. Agric. Syst. 70 (2–3), 421–443.
Keating, B.A., Carberry, P.S., Hammer, G.L., Probert, M.E., Robertson, M.J., Holzworth, D., Huth, N.I., Hargreaves, J.N.G., Meinke, H., Hochman, Z., McLean, G., Verburg, K., Snow, V., Dimes, J.P., Silburn, M., Wang, E., Brown, S., Bristow, K.L., Asseng, S., Chapman, S., McCown, R.L., Freebairn, D.M., Smith, C.J., 2003. An overview of APSIM, a model designed for farming systems simulation. Eur. J. Agron. 18, 267–288.
Keating, B.A., Meinke, H., Probert, M.E., Huth, N.I., Hills, I.G., 2001. Nwheat: Documentation and Performance of a Wheat Module for APSIM. Tropical Agronomy Memorandum, CSIRO Division of Tropical Agriculture, 306 Carmody Rd, St. Lucia, Qld 4067, Australia.
Lawless, C., Semenov, M.A., Jamieson, P.D., 2005. A wheat canopy model linking leaf area and phenology. Eur. J. Agron. 22 (1), 19–32.
Manschadi, A.M., Hochman, Z., McLean, G., DeVoil, P., Holzworth, D., Meinke, H., 2006. APSIM-Barley model – adaptation of a wheat model to simulate barley growth and development. In: Turner, N.C., Acuna, T. (Eds.), 13th Australian Agronomy Conference: Perth, Western Australia.
Martin, R.J., Jamieson, P.D., Gillespie, R.N., Maley, S., 2001. Effects of Timing and Intensity of Drought on the Yield of Oats (*Avena sativa* L.). Australian Agronmy Conference, Hobart.
Martin, R.J., Sinton, S.M., Jamieson, P.D., Sonego, M., 1998a. The Effect of Daylength on Final Leaf Number in Drummond Oats. Australian Agronomy Conference, Wagga Wagga.
Martin, R.J., Sinton, S.M., Jamieson, P.D., Sonego, M., 1998b. The Effect of Temperature on Leaf Apperance Rate in Drummond Oats. Australian Agronomy Conference, Wagga Wagga.
Meinke, H., Hammer, G.L., van Keulen, H., Rabbinge, R., Keating, B.A., 1997. Improving wheat simulation capabilities in Australia from a cropping systems perspective: water and nitrogen effects on spring wheat in a semi-arid environment. Eur. J. Agron. 7, 75–88.
Meinke, H., Hammer, G.L., Want, P., 1993. Potential soil water extraction by sunflower on a range of soils. Field Crops Res. 32, 59–81.
Moore, A.D., Holzworth, D.P., Herrmann, N.I., Huth, N.I., Robertson, M.J., 2007. The Common Modelling Protocol: a hierarchical framework for simulation of agricultural and environmental systems. Agric. Syst. 95, 37–48.
Peake, A., Whitbread, A., Davoren, B., Braun, J., Limpus, S., 2008. The development of a model in APSIM for the simulation of grazing oats and oaten hay. In: Unkovich, M. (Ed.), "Global Issues. Paddock Action." Proceedings of 14th Agronomy Conference: Adelaide, South Australia.
Penning de Vries, F.W.T., Jansen, D.M., Bakema, A., Ten Berge, H.F.M., 1989. Simulation of Ecophysiological Processes of Growth in Several Annual Crops. Pudoc, Wageningen, Netherlands.
Probert, M.E., Dimes, J.P., Keating, B.A., Dalal, R.C., Strong, W.M., 1998a. APSIM's water and nitrogen modules and simulation of the dynamics of water and nitrogen in fallow systems. Agric. Syst. 56 (1), 1–28.
Probert, M.E., Robertson, M.J., Poulton, P.L., Carberry, P.S., Weston, E.J., Lehane, K.J., 1998b. Modelling lucerne growth using APSIM. In: Proceedings of the Ninth Australian Agronomy Conference, Wagga Wagga, pp. 247–250.
Rahman, J.M., Seaton, S.P., Cuddy, S.M., 2004. Making frameworks more useable: using model introspection and metadata to develop model processing tools. Environ. Model. Softw. 19 (3), 275–284.
Ritchie, J.T., Otter, S., 1985. Description and Performance of CERES-wheat. A User-oriented Wheat Model, ARS Wheat Project, pp. 159–176.
Robertson, M.J., Carberry, P.S., 1998. Simulating growth and development of soybean in APSIM. In: Proceedings Tenth Australian Soybean Conference, : Brisbane, 15-17 September, pp. 130–136.
Robertson, M.J., Carberry, P.S., Huth, N.I., Turpin, J.E., Probert, M.E., Poulton, P.L., Bell, M., Wright, G.C., Yeates, S.J., Brinsmead, R.B., 2002. Simulation of growth and development of diverse legume species in APSIM. Aust. J. Agric. Res. 53, 429–446.
Robertson, M.J., Fukai, S., Ludlow, M.M., Hammer, G.L., 1993. Water extraction by grain sorghum in a sub-humid environment. II. Extraction in relation to root growth. Field Crops Res. 33, 99–112.
Robertson, M.J., Sakala, W., Benson, T., Shamudzarira, Z., 2005. Simulating response of maize to previous velvet bean (Mucuna pruriens) crop and nitrogen fertiliser in Malawi. Field Crops Res. 91 (1), 91–105.
Snow, V., Huth, N., 2004. The APSIM – Micromet module. HortResearch, pp. 18.
Snow, V.O., Houlbrooke, D.J., Huth, N.I., 2007. Predicting soil water, tile drainage, and runoff in a mole-tile drained soil. N. Z. J. Agric. Res. 50 (1), 13–24.
Sonego, M., Moot, D.J., Jamieson, P.D., Martin, R.J., Scott, W.R., 2000. Apical development in oats predicted by leaf stage. Field Crops Res. 65 (1), 79–86.
Stockle, C.O., Donatelli, M., Nelson, R., 2003. CropSyst, a cropping systems simulation model. Eur. J. Agron. 18 (3–4), 289–307.
Teixeira, E.I., Brown, H.E., Meenken, E.D., Moot, D.J., 2011. Growth and phenological development patterns differ between seedling and regrowth lucerne crops (Medicago sativa L.). Eur. J. Agron. 35 (1), 47–55.
Teixeira, E.I., Moot, D.J., Brown, H.E., 2009. Modeling seasonality of dry matter partitioning and root maintenance respiration in lucerne (Medicago sativa L.) crops. Crop Pasture Sci. 60 (8), 778–784.
Teixeira, E.I., Moot, D.J., Mickelbart, M.V., 2007a. Seasonal patterns of root C and N reserves of lucerne crops (Medicago sativa L.) grown in a temperate climate were affected by defoliation regime. Eur. J. Agron. 26 (1), 10–20.
Teixeira, E.I., Moot, D.J., Pollock, K.J., Brown, H.E., 2007b. How does defoliation management affect yield, canopy forming processes and light interception in lucerne (Medicago sativa L.) crops? Eur. J. Agron. 27 (1), 154–164.
Thornby, D., Walker, S.R., Whish, J.P.M., 2006. Simulating Weed Persistence and Herbicide Resistance in the Northern Grain Region Using a Validated Crop Growth Model with Extensions for Seedbank Dynamics and Mating. Weed Management Society of South Australia, Victoria, Australia.
Turpin, J.E., Robertson, M.J., Haire, C., Bellotti, W.D., Moore, A.D., Rose, I., 2003. Simulating fababean development, growth, and yield in Australia. Aust. J. Agric. Res. 54 (1), 39–52.
van Keulen, H., Penning de Vries, F., Drees, E., 1982. 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. Centre for Agricultural Publishing and Documentation, Wageningen, Netherlands.
Verburg, K., Bond, W.J., Hirth, J.R., Ridley, A.M., 2007. Lucerne in crop rotations on the Riverine Plains. 3*. Model evaluation and simulation analyses. Aust. J. Agric. Res. 58 (12), 1129–1141.
Wang, E., Robertson, M.J., Hammer, G.L., Carberry, P.S., Holzworth, D., Meinke, H., Chapman, S.C., Hargreaves, J.N.G., Huth, N.I., McLean, G., 2003. Development of a generic crop model template in the cropping system model APSIM. Eur. J. Agron. 18, 121–140.
Wang, E., Smith, C.J., Bond, W.J., Verburg, K., 2004. Estimations of vapour pressure deficit and crop water demand in APSIM and their implications for prediction of crop yield, water use, and deep drainage. Aust. J. Agric. Res. 55 (12), 1227–1240.
Wang, E.L., Engel, T., 2000. SPASS: a generic process-oriented crop model with versatile windows interfaces. Environ. Model. Softw. 15 (2), 179–188.
Whish, J.P.M., Carberry, P.S., Robertson, M.J., Smith, P., 2002. The Development of a Wild Oat Simulation Model for APSIM. Plant Protection Society of Western Australia Inc, Victoria Park, Australia.

Further reading

Huth, N., Banabas, M., Nelson, P., Webb, M., 2013. Lessons from extending and existing agricultural systems model to simulate oil palm. Environ. Model. Simul. (in this issue).