Overview of the "*Sirius-BioMa-MaizeLAI-Component"*

1. Content of the "*Sirius-BioMa-MaizeLAI-Component*" folder

   The folder "*Sirius-BioMa-MaizeLAI-Component*" contains:

   - ➢ The source code of the *SQ-MaizeLAI BioMa* Component (*SiriusQuality-MaizeLAI* folder), see *Sirius-BioMa-MaizeLAI-Component* document
   - ➢ The BioMa dlls which are mandatory to use the component (*BioMa-DLL* folder)
   - ➢ A console application which provides an example for the use of the component (*SiriusQuality-MaizeLAIConsole* folder)
   - ➢ Unit tests for the component (*UnitTestMaizeLAI* folder)
   - ➢ A visual studio solution which allows to run the console application and the unit tests (*SiriusQuality-MaizeLAI.sln*)
   - ➢ A detailed documentation about both the calculation scheme and the equations of the component (*Documentation* folder)

2. How to use the component:

   The *MaizeLAI* component can be added to a *BioMa* solution with the *CLIC* tool or be plugged to your model via a wrapper. Here the composition with *CLIC* will not be presented (see instead *BioMa* solution documentation). In what follows, we will first make a quick overview of the component, then we will present the wrapper and the leaf layer classes of *SiriusQuality*. Finally, we will explain how they are used via a console application.

   a. Overview of the component

      The component (*Sirius-BioMa-MaizeLAI-Component)* contains six folders:

      - ➢ The *domainClass* folder. Here can be found:
        - The getter and setter of the general states (*MaizeLAIState.cs*) and those for the states

specific to the leaf layers (*MaizeLeafState.cs*). The state of the leaf layers are *Lists.*
        - The metadata on the states (*MaizeLAIStateVarInfo.cs and MaizeLeafStateVarInfo.cs*)

    - The *Strategies* folder where the simple strategies and the composite one (*MaizeLAI.cs*) can be found. The composite strategy allows to call sequentially the simple strategies via its *Estimate* function or the *UpdateLeafArea* simple strategy via the function named the same
    - The *API* folder containing the classes for the Application Programming Interface
    - The *Utilities* folder that contains basic Math functions which can be used anywhere in the strategies
    - An *XML* folder where the xml files used to generate strategies and domain classes with the BioMa tools can be found
    - The *obj/Debug* folder containing the dll of the component after having built it and the *BioMA* dlls which are mandatory for the project
    - The *bin* folder for binaries

b. Maize LAI wrapper

The wrapper (*SiriusQuality-MaizeLAIConsole / MaizeLAIWrapper.cs*) makes possible:

- The initial loading of the parameters
- The connection of the *SiriusQuality* leaf layer objects to the leaf layers of the component.
- The day by day valorization of the inputs
- The daily call of the component (the *Estimate* function of the composite class and the *UpdateLeafAreas* functions are called separately in the core of *SiriusQuality* via the wrapper)
- And the daily export of the outputs

For these purposes an object *MaizeLeafState* along with two *MaizeLeafState* ones (*MaizeLeafState_* and

*MaizeLeafState1_*) are instantiated. Sometime, *MaizeLeafState_* is used as an input for the component. Then it contains the information of the day before. *MaizeLeafState1_* is used as an output. These objects allow to valorize inputs and export outputs (via getter). In addition, an object of the composite class is instantiated (*MaizeLAI_*). It helps to valorize the parameters and call the *Estimate* or the *UpdateLeafAreas* functions of the composite class.

The valorization of the parameters is done in the constructor of the wrapper object via the *loadParametersMaize* function. The *Estimate* and *UpdateArea* functions can be called everywhere in the code. their arguments are the values of the input for the current day. Four steps are achieved in these functions:

- The current day inputs are valorized
- The information on the leaf layer object in *SiriusQuality* are loaded in the leaf layer lists of the component via the *FillIntputLayersMaize* function
- The *Estimate* function or the *UpdateLeafAreas* functions of the composite class are called
- The information on the leaf layer object of *SiriusQuality* are updated from the new values of the leaf layer lists which are now outputs of the component (*FillOutputLayersMaize*)

A last function has to be mentioned: *CreateLeafLayerLAIComponentMaize* is used to Add a null element to the layer list of the component each time a leaf layer is created in *SiriusQuality*.

c. Leaf layer classes of *SiriusQuality*

Two classes are used to model the leaf layers in *SiriusQuality.* They were copied in the folder *SiriusQuality-MaizeLAIConsole*:

- *LeafState.cs* contains only a public object of type *enum* which is use to enumerate and store the possible states of a leaf layer (Growing, Mature)

    - *LeafLayer.cs* allows creating as many leaf layer objects as grown in the canopy via a constructor. Here can also be found a copy constructor. The class proprieties (*GAI*, *laminaAI*, *State*, *DeltaDM*, *DeltaAI*…) characterize each individual leaf layer. The *LeafLayer* objects can be seen as containers for the information on single leaf layers.

d. Console application

The *SiriusQuality-MaizeLAIConsole/Program.cs* class can be divided in five parts:

- Inputs that cannot be calculated are grouped in arrays. The count of the arrays corresponds to the number of days in the simulation
- An object *MaizeLAIWrapper* is instantiated to be able to call the wrapper’s *Estimate* and *UpdateAreas* functions and export outputs
- When entering the loop over the days of the plant cycle several steps are achieved daily:
    - Other inputs are calculated
    - Leaf layer are created both at the console application level (*LeafLayer.cs* class) and in the *MaizeLAI* component (*CreateLeafLayerLAIComponentMaize*)
    - The *MaizeLAI* component is called via the *Estimate* function of the wrapper
    - The area index of the leaf layers is updated via the *UpdateAreas* function of the wrapper (which calls the *UpdateLeafAreas* function of the component)
    - The total Green Area Index of the canopy is updated (increase for growing leaves)
    - The biomass of individual leaves is updated accordingly and that of the whole canopy, as well.
- Outputs of the component are called from the wrapper and printed

3. List of the provided libraries

   Six libraries are mandatory to be able to run the component. Five of them are part of the *BioMa* framework and are loaded both in the component and in the console application projects, while *CRA.AgroManagement2014.dll*, *CRA.AgroManagement2014.Impacts.dll* and *CRA.AgroManagement2014.dll* are used for the management of agronomic events (irrigation, fertilization…), *CRA.Core.Preconditions.dll* is used for the test of the value of the parameters, inputs and outputs and *CRA.ModelLayer.dll* is dedicated to the generation of domain classes and strategies.

   When building the *SiriusQuality-MaizeLAI* project a library called *SiriusQuality-MaizeLAI.dll* is created. It is loaded in the console application project (and the corresponding includes are added on top of the wrapper class).

   These six libraries have to be loaded in any project aiming at working with the *Maize-LAI* component.

4. List of inputs and outputs of the component:

Inputs		
Name in the code	units	Definition
`newLeafHasAppeared`	-	1: a new leaf has appeared the day before, 0: if not
`finalLeafNumber`	leaf	Decimal final leaf number on main stem
`leafNumber`	leaf	Decimal Leaf number on main stem the current day
`previousLeafNumber`	leaf	Decimal Leaf number on main stem the day before
`FPAW`	-	Fraction of available water
`cumulTTPhenoMaize`	°Cd	Thermal time cumulated since sowing date
`deltaTTPhenoMaize`	°Cd	Daily growth thermal time increase of the day
`VPDairCanopy`	hPa	Air-Canopy Vapor Pressure Deficit
`TCanopyHourly`	°C	Array of 24 values for the canopy temperature during the day
`VPDeq`	hPa	Equivalent Air-Leaf Vapor Pressure deficit taking into account Photosynthetically Active Radiation effects
`radIntercepted`	MJ/m^2(leaf)	Radiation intercepted by the canopy
`GAI`	m^2(leaf)/m^2(ground)	List which stores the Green Area Index (Lamina Area Index) of the day before for each leaf layer
`cumIntRad`	MJ/m^2(leaf)	Cumulated intercepted radiation the day before

Outputs		
Name in the code	units	Definition
`State`	-	List of the state of each leaf layer as an *integer* (0: Growing, 1:Mature). The MaizeLAIWrapper's function `getLeafStateList() convert the list of integers in a list of` *`LeafState.`*
`potentialIncDeltaArea`	m^2(leaf)/m^2(ground)	Total daily potential expansion rate (without Nitrogen availability effects). It corresponds to the sum of the potential expansion rate of each layer.
`incDeltaArea`	m^2(leaf)/m^2(ground)	Total daily expansion rate under drought and nitrogen stress.
`WaterLimitedPotDeltaAIList`	m^2(leaf)/m^2(ground)	List which stores the expansion under drought stress for each leaf
`MaxAI`	°Cd	List which stores the maximum Green Area Index attained the current day for each leaf layer
`DeltaAI`	m^2(leaf)/m^2(ground)	List which stores the daily expansion rate under drought and nitrogen stress for each leaf layer
`laminaAI`	m^2(leaf)/m^2(ground)	List which stores the lamina area index under drought and nitrogen stress for each leaf layer the current day
`length`	mm	List which stores the individual length of leaves under drought stress
`width`	mm	List which stores the individual length of leaves under drought stress
`area`	mm^2	List which stores the individual area of leaves under drought stress
`exposedArea`	mm^2	List which stores the individual exposed area of leaves under drought stress

A list of the parameters dedicated to the calculations of the component is given in Table A2 of */Documentation/SQ3_WheatLAI_component.pdf*.