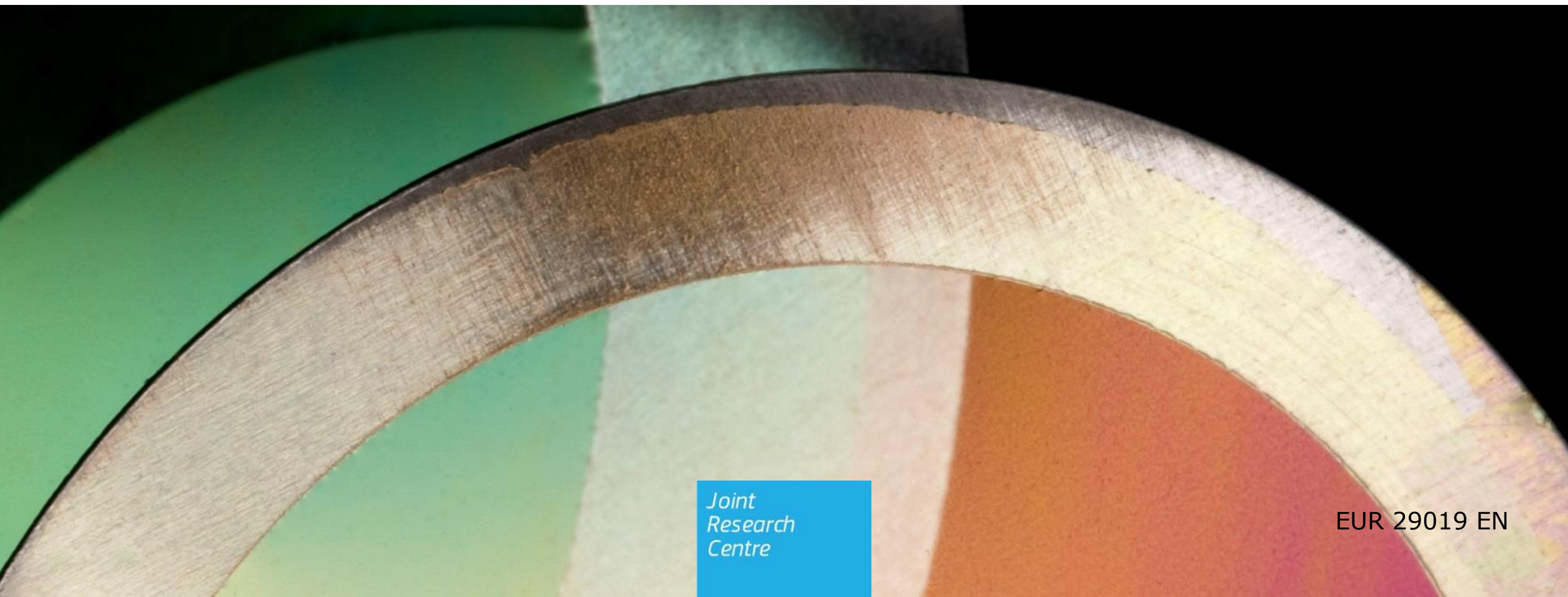
European Commission

JRC TECHNICAL REPORTS

Bioma platform advancements during 2017

Fumagalli, Davide

Niemeyer, Stefan



Joint Research Centre

EUR 29019 EN

JRC Science Hub
https://ec.europa.eu/jrc

JRC110317

EUR 29019 EN

PDF ISBN 978-92-79-77324-2 ISSN 1831-9424 doi:10.2760/713904

Luxembourg: Publications Office of the European Union
© European Union, 2017

How to cite this report: Fumagalli, Niemeyer, *Bioma platform advancements during 2017 - JRC D5 Food Security Unit,* EUR 29019 EN, Publications Office of the European Union, Luxembourg, 2017, ISBN 978-92-79-77324-2, doi:10.2760/713904, JRC110317

Contents

Abstract

In this report we describe the advancements on the Bioma Framework developed during year 2017. Given that the Bioma platform is quite mature, its core was not recently changed. So that the majority of changes concerns the implementation of the models developed in the platform. Moreover, during 2017 we also set up an alternative version of the framework itself, based on a new developing framework called .NET Core, with the purpose of being able to create a version of Bioma runnable on Linux.

Therefore this document is organized in two chapters: the advancements on the models and the creation of the new version of the platform.

Bioma is a framework for develop and run agronomical models. The Bioma framework is used in the context of unit D5 since many years and, starting from year 2015, it is used also in the operational chain for the Agri4Cast bulletin. The changes described in chapter 1 apply also to the operational use of Bioma, whereas the content of chapter 2 does not have, for now, an impact on the activities of the unit.

The documentation of Bioma and of the other software cited in this document are in the Reference chapter.

1 Advancements on the models implemented in the framework

1.1 Water balance bug solved in Wofost water balance component

During 2017 we identified and solved a bug regarding the soil water balance component user in the operational version. This component is used, together with the Wofost crop model, to simulate the water available for the crop and the water limited yield of the crop. The component was implemented following the algorithm described in the Alterra/WER's Wofost documentation.

We discovered the bug while making a comparison between the Alterra/WER PCSE software (which is a python implementation of the same modelling solution) and the Bioma version of the Wofost modelling solution.

1.1.1 Bug summary

There is a difference between the soil water calculation implementation between the Alterra/WER's PCSE version of Wofost and the Bioma version of Wofost.

The difference is on the calculation of the daily increase of water in the rooted zone. One of the possible increases of the water in the rooted zone is the fact that the roots grow, and so they intercept more water. The quantity of water to add to the rooted zone is the daily increase of depth multiplied by WLOW (quantity of water in the unrooted layer). This quantity can be seen as the magnitude of the movement of water from unrooted to rooted zone.

This is the exact formula:

Δ WROOT = RATE OF INCREASE OF WATER IN THE ROOTED ZONE = WLOW * (NEW ROOT DEPTH − ROOT DEPTH OF DAY BEFORE)/(MAX ROOT DEPTH − ROOT DEPTH OF DAY BEFORE)

WHERE

WLOW = QUANTITY OF WATER IN THE UNROOTED LAYER

In PCSE this calculation happens before the update of WLOW for the current day, whereas in Bioma this calculation happens after the update of WLOW.

The WLOW during the first phase of the growing season is usually decreasing day by day; So, generally, the WLOW used by Bioma is lower. It can be easily seen in the explanatory table below: in this example we imagine the root depth increases every day by 2 cm. In the Δ WROOT column it is shown the value of the calculation of rate of increase of water in the rooted zone for each day. It can be seen that Δ WROOT increases day by day. So, if the previous day calculation is erroneously used instead of the current day calculation the quantity of water in rooted zone would be less than the correct value. This is what happens in the bugged version of the model: less water is added to the rooted zone and the bugged simulations are in general 'drier' than the correct ones.

#DAY	old root depth	new root depth	Max root depth	Δ Wroot
1	0	2	100	2/100 = 0.0200
2	2	4	100	2/98 =0.0204
3	4	6	100	2/96 =0.0208

4	6	8	100	2/94 =0.0212

Figure 1 - An example of the calculation of increase of water in rooted zone. We imagine root depth daily increase is constant and equal to 2 cm.

The difference on the water in rooted zone used to propagate also on the other variables implied in the water balance, causing a not correct increase of transpiration and loss of water to subsoil.

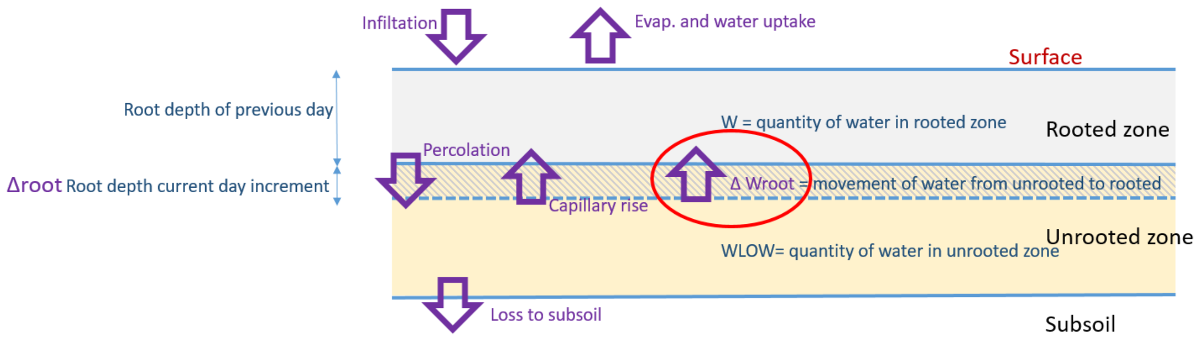

Infiltation
Evap. and water uptake
Surface
Root depth of previous day
W = quantity of water in rooted zone
Rooted zone
Percolation
Δroot Root depth current day increment
Δ Wroot = movement of water from unrooted to rooted
Capillary rise
Unrooted zone
WLOW= quantity of water in unrooted zone
Loss to subsoil
Subsoil

Figure 2 - Schema of the water balance. The rate of increase of water in the rooted zone is highlighted in red (ΔWroot)

1.1.2 Effects on the simulation results

The bug affected simulations introduced an artificial water stress and so the water limited yields calculated were lower than the correct values. The magnitude of the difference depends on the crop and on the simulated area, being less important for places and crops were the water stress is less frequent and important.

To estimate the impact of the change in the European bulletin simulation results, we performed the simulation of the whole Europe, all the crops, and confronted the results to the simulation run without the bug correction. The comparison was done at STU level to catch entirely the differences in the soil water calculation. Only year 2016 was considered. The total number of combinations CROP/GRID/STU for the European bulletin is circa 1.8 million.

In the following graphs it is shown the magnitude of the error in terms of percentage difference between the PCSE correct results and the Bioma wrong results.

On the X axis it is shown the percentage difference calculated as

Percentage difference = PCSE results - Bioma results / PCSE results

On the Y axis it is shown the number of locations that experienced the difference.

The area below the curve can be seen as the magnitude of the error.

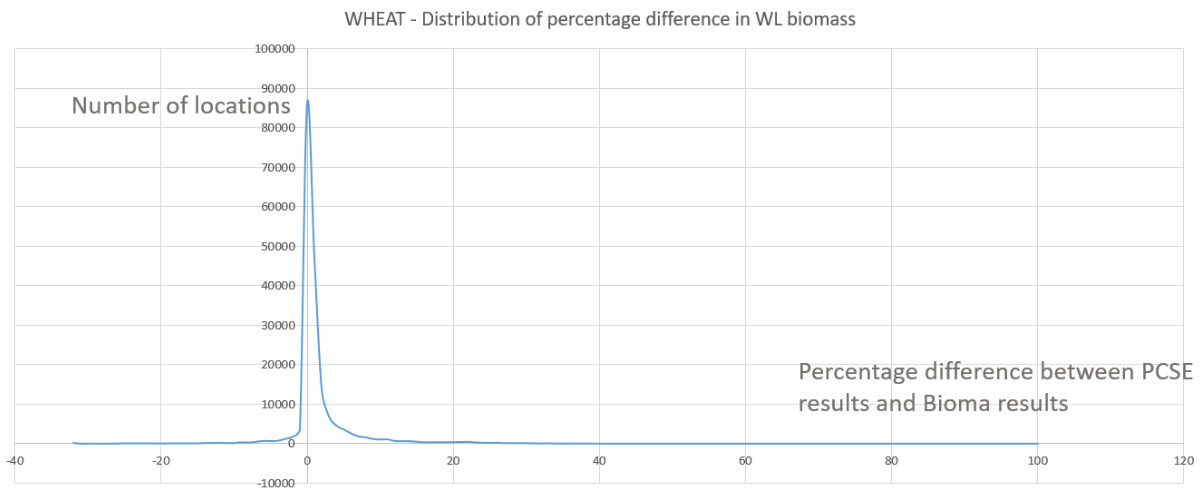


Figure 3 - Distribution of percentage difference in water limited biomass for wheat in Europe

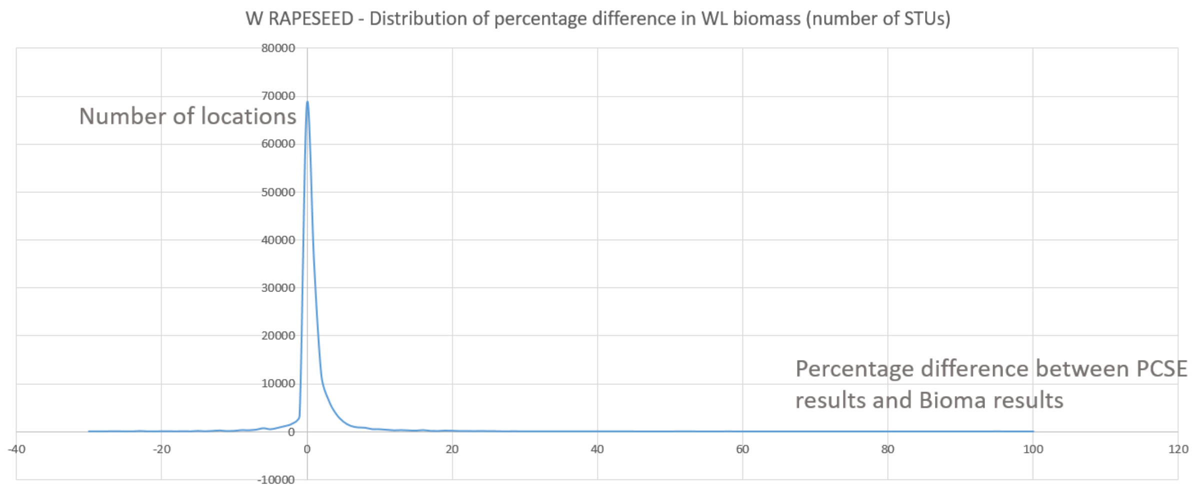


Figure 4 - Distribution of percentage difference in water limited biomass for winter rapeseed in Europe

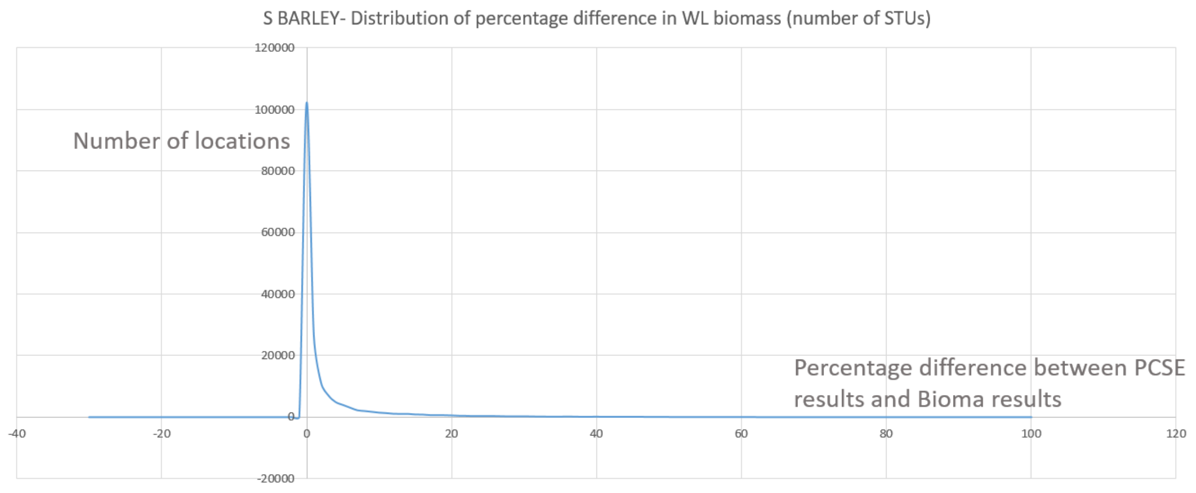


Figure 5 - Distribution of percentage difference in water limited biomass for spring barley in Europe

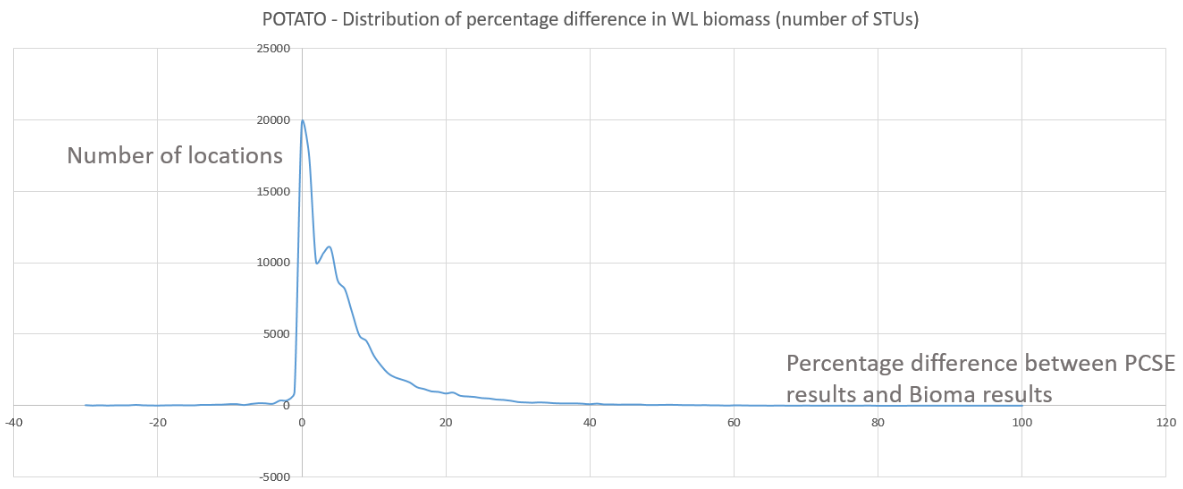


Figure 6 - Distribution of percentage difference in water limited biomass for potato in Europe

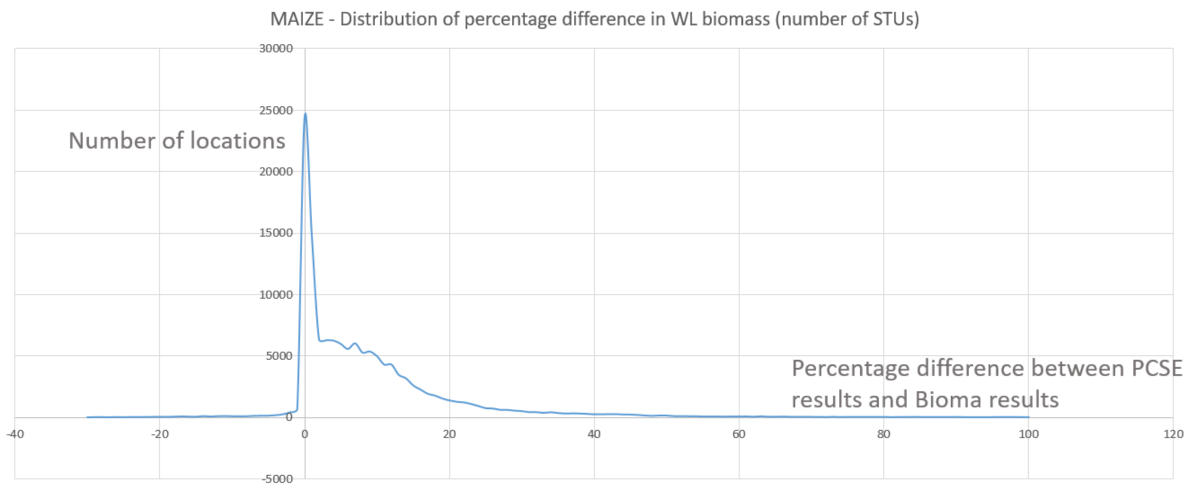


Figure 7 - Distribution of percentage difference in water limited biomass for maize in Europe

From the figures above it can be seen that the crops most affected by the problem were the summary crops (maize and potato) where the water stress is more frequent and where its sensitivity on the final yield is higher. For wheat, sunflower, rapeseed, barley, rye and beans the differences are in general below 5%. For maize, potato and sugar beet there is a significant number of locations where the difference is greater than 5%, up to 20%.

The geographic distribution of the errors is shown in the next figure for maize. The majority of the errors happened where the climate is drier and so the water limitation is in general higher.

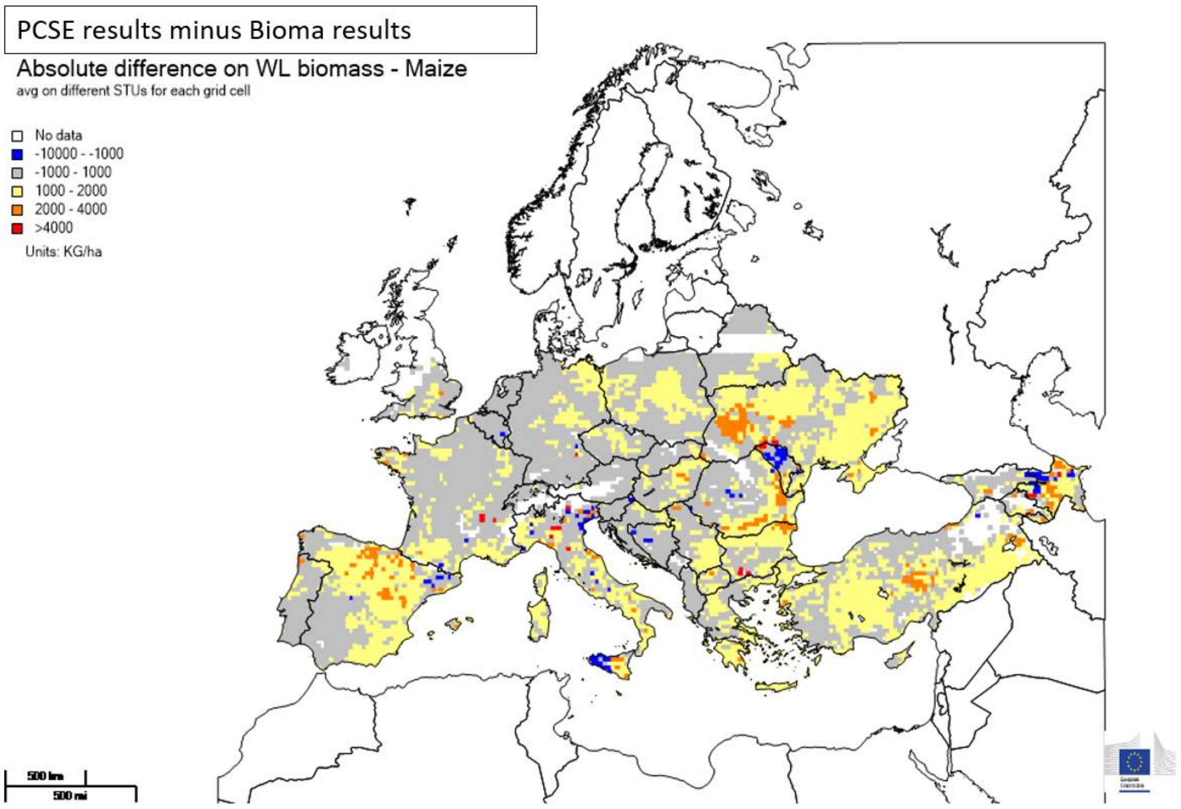

PCSE results minus Bioma results
Absolute difference on WL biomass - Maize
avg on different STUs for each grid cell
□ No data
■ -10000 - -1000
■ -1000 - 1000
■ 1000 - 2000
■ 2000 - 4000
■ >4000
Units: KG/ha
500 km
500 mi

Figure 8 - Geographical distribution of the errors on water limited biomass for maize, in absolute values (KG/ha)

1.1.3 Correction applied

The order of the calculations of WLOW and rate of increase of water in the rooted zone was reversed.

1.1.4 Tests to check the correction

Many tests were performed to check that, after the correction was applied, the results of PCSE and Bioma coincide. All the tests gave positive results. Here we report the results of one of these tests.

The test was performed for a particular cell/year in which the discrepancy between the two versions was big. Window: RUK (Russia and Kazakhstan), cell number: 2055090, STU number: 1215, Year: 2007, Crop: maize.

By applying the described correction, the Bioma results coincide to the PCSE results, as can be seen in the next figures (The last part of each season is very different but not important: Bioma keeps crop 'alive' until December, whereas the PCSE simulation stops crop just after maturity, which happens at time step number 118).

Only remains a little discrepancy in the final value of the water limited biomass, probably due to some other factor not yet considered.

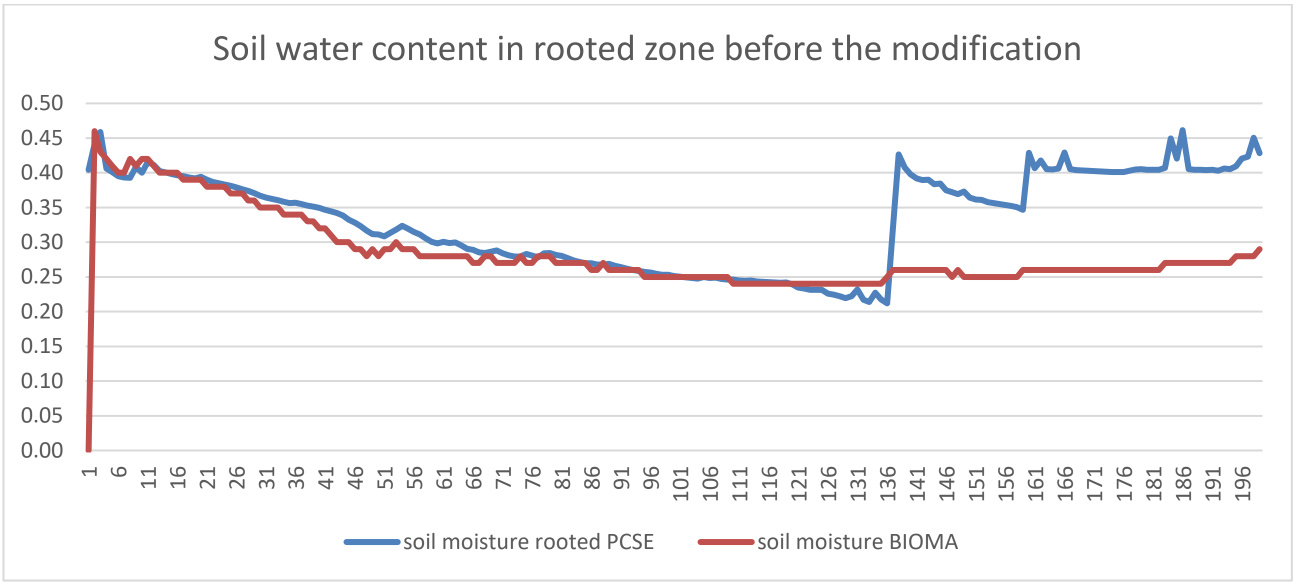


Figure 9 - Differences in soil moisture in rooted zone in the test case between PCSE and Bioma before the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the soil water content in m^3/M^3

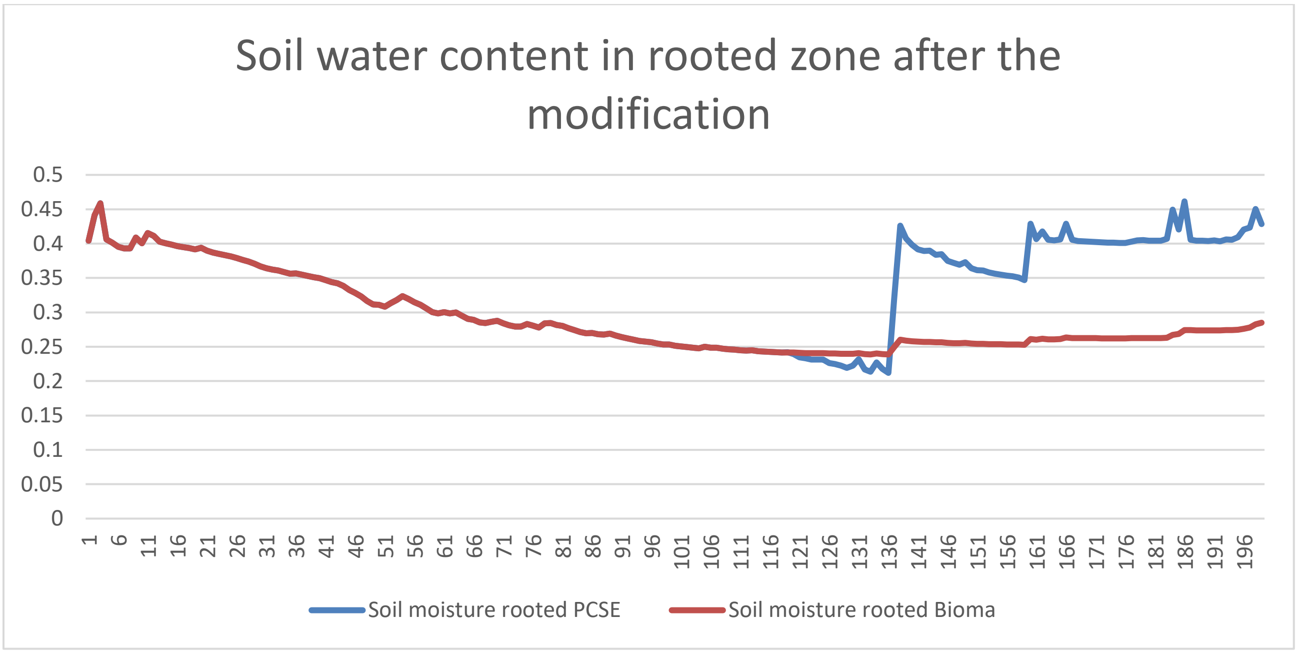


Figure 10 - Differences in the soil moisture in rooted zone in the test case between PCSE and Bioma after the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the soil water content in m^3/M^3

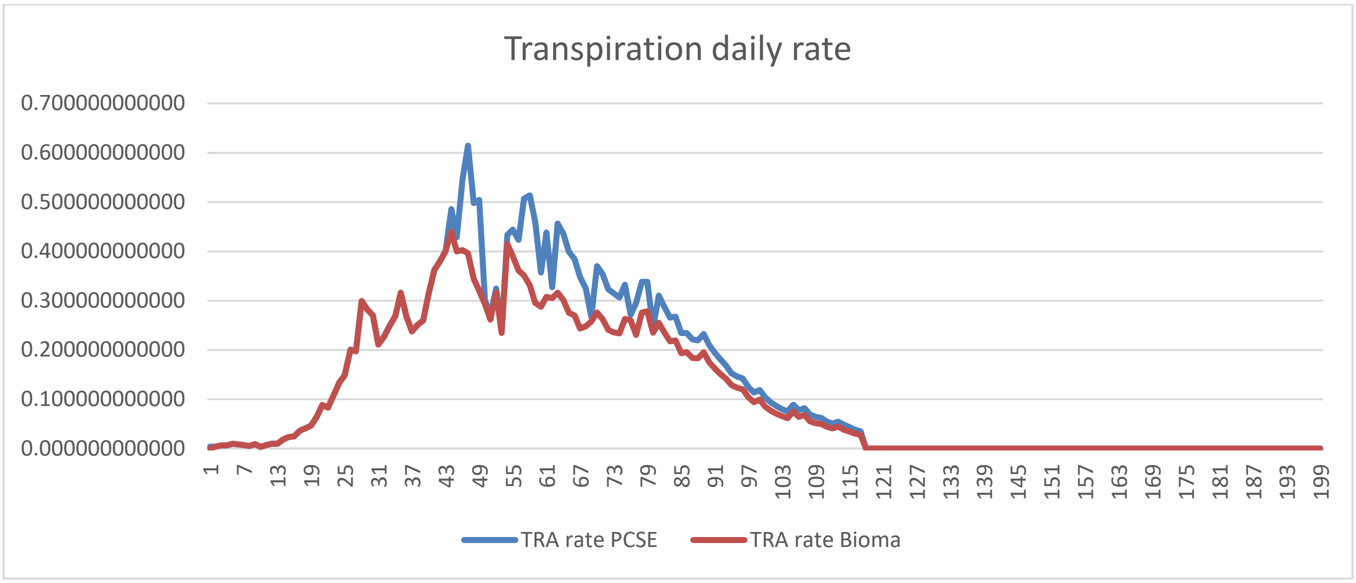


Figure 11 - Differences in the transpiration daily rate in the test case between PCSE and Bioma before the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the transpiration in cm

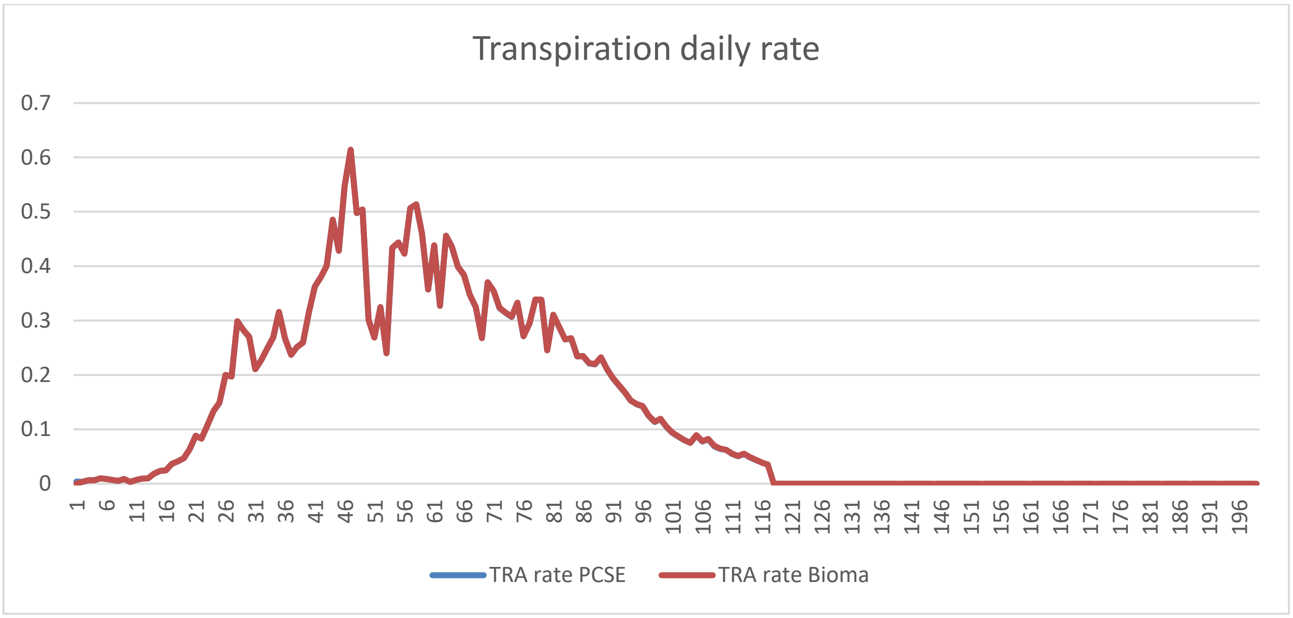


Figure 12 - Differences in the transpiration daily rate in the test case between PCSE and Bioma after the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the transpiration in cm

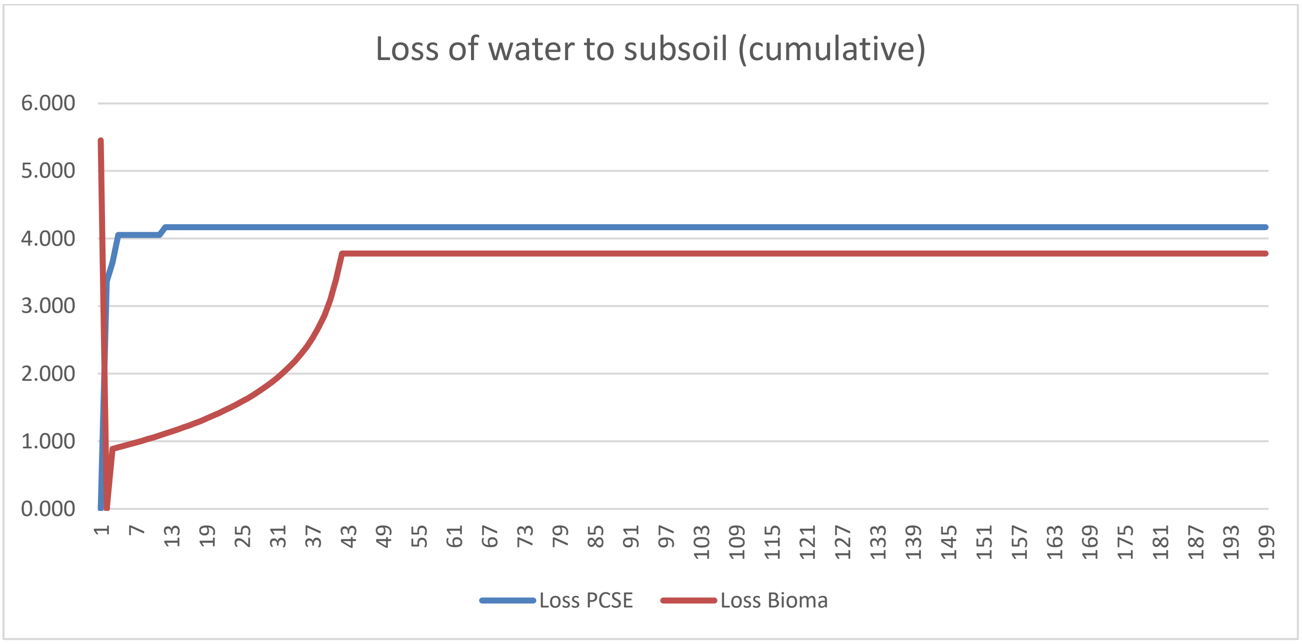


Figure 13 - Differences in the loss of water to subsoil in the test case between PCSE and Bioma before the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the loss of water in cm

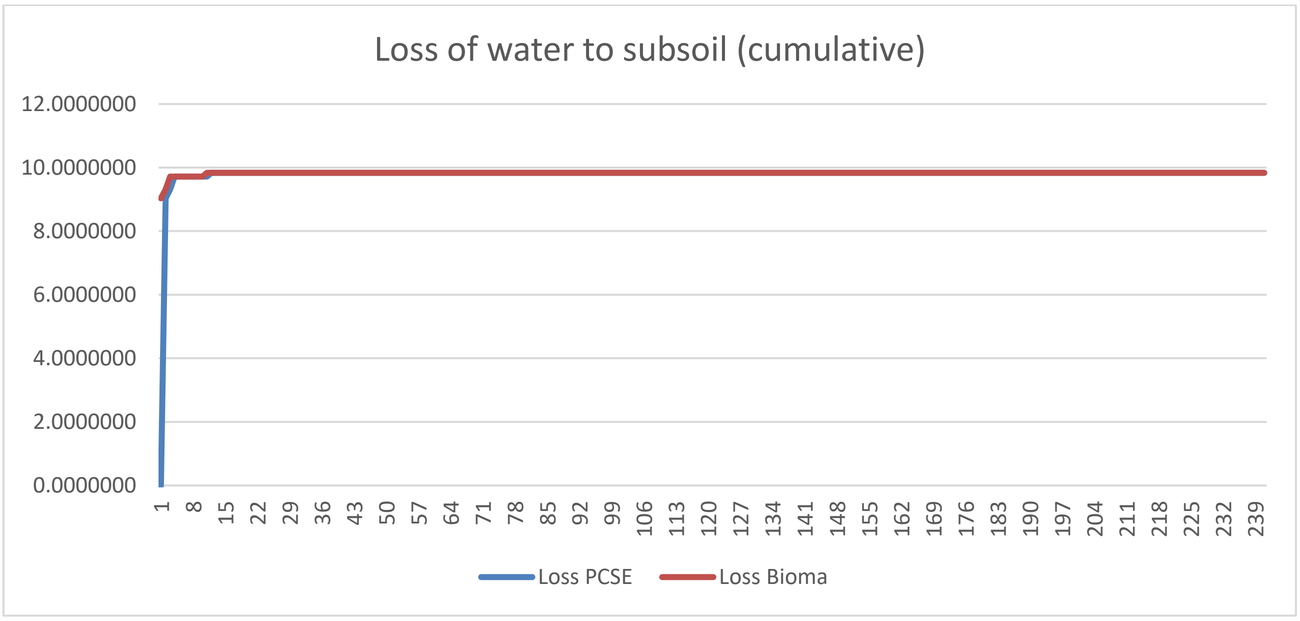


Figure 14 - Differences in the loss of water to subsoil in the test case between PCSE and Bioma after the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the loss of water in cm

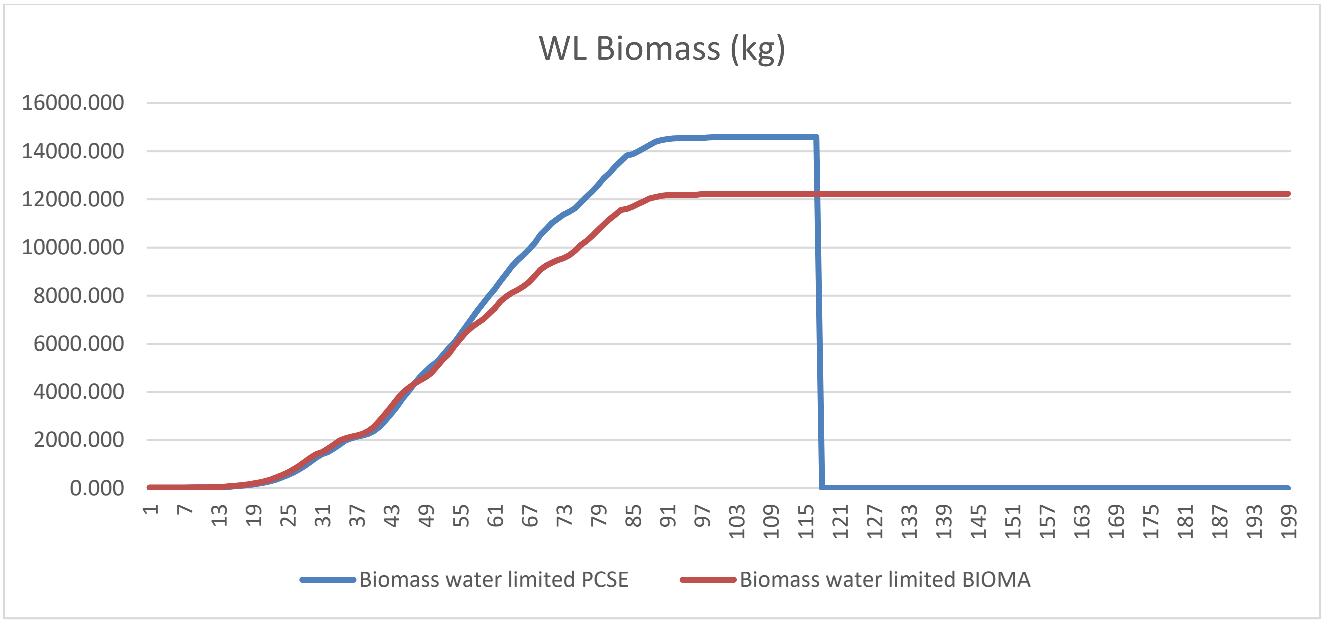


Figure 15- Differences in the loss of water to subsoil in the test case between PCSE and Bioma before the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the water limited biomass in KG/ha.

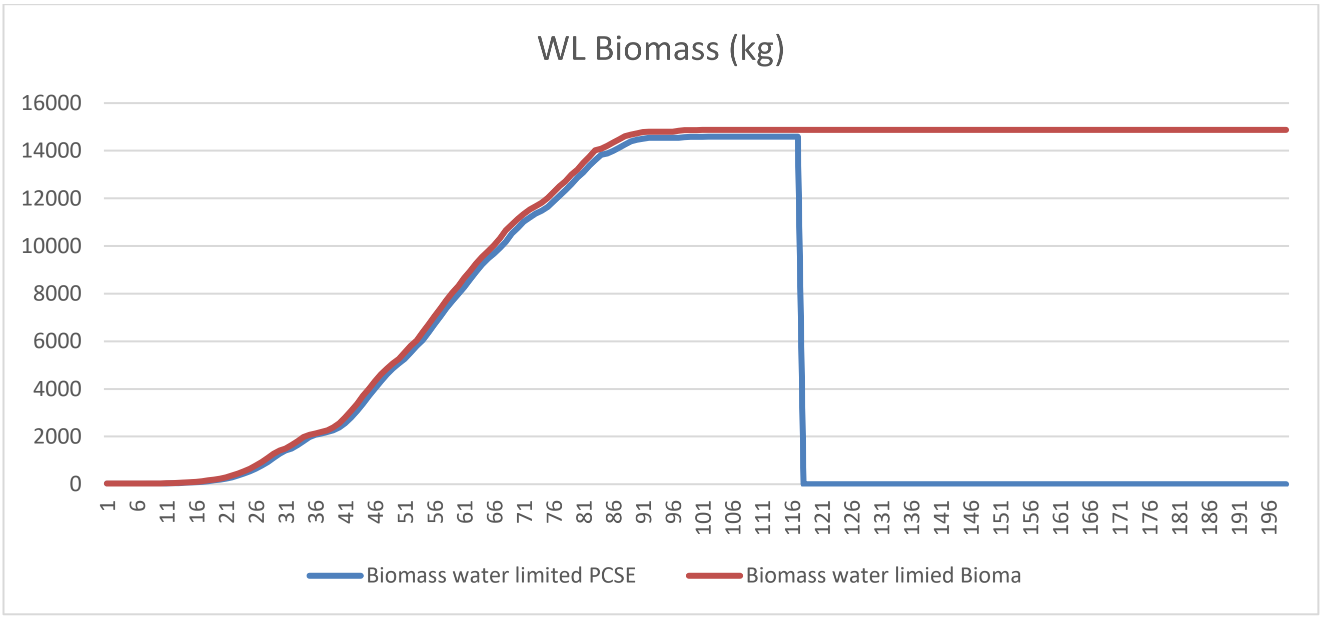


Figure 16- Differences in WL biomass in the test case between PCSE and Bioma after the bug correction was applied. On the X axis it is shown the Julian day, on the Y axis it is shown the water limited biomass in KG/ha.

1.1.5 Further steps

The test results have been accepted, so it was planned to introduce the correction in the operational system starting from the beginning of 2018 bulletin season.

1.2 Modifications to the Bioma software for simulating vernalized winter wheat

This chapter describes the modifications applied to the Bioma system to manage the simulation of the vernalized winter wheat. The particularity of this crop is that it is a cross annual crop, meaning that it is sown in autumn and harvested in spring. This is the first crop with this behaviour simulated in the operational environment, so it was necessary to make some modifications to the system.

The vernalized wheat simulations were run during 2017 by Alterra/WER on the operational system. To make it possible, we provided an updated release of Bioma (operational console version) to Alterra/WER and we gave them support in using it.

The main release of Bioma (provided on 02/02/2017) included the improvements to the model to properly simulate vernalized winter wheat:

- The Bussel algorithms for simulating the vernalization was included in Wofost.
    - The algorithm is the same described in the PCSE documentation of class "Vernalisation" (http://pcse.readthedocs.io/en/stable/_modules/pcse/crop/phenology.html)
    - The four Bussel algorithm parameters were added in the list of Wofost parameters managed by Bioma and read from the DB:

      VERNSAT: SATURATED VERNALISATION REQUIREMENTS IN DAYS

      VERNBASE: BASE VERNALISATION REQUIREMENTS IN DAYS

      VERNRTB: RATE OF VERNALISATION AS A FUNCTION OF DAILY MEAN TEMPERATURE.

      VERNDVS: CRITICAL DEVELOPMENT STAGE AFTER WHICH THE EFFECT OF VERNALIZATION IS HALTED
    - The call to the Bussel algorithm was added in the Bioma Wofost "PotentialPhenologyC" strategy class. The call is subordinate to the value of the IDSL parameter.
- It was introduced the crop parameter IDSL for switching on and off the photoperiod and vernalization effects. The IDSL parameter was already present in the operational CGMS database, but Bioma never used it, since neither photoperiod not vernalization were previously considered. IDSL can assume these values:
    - IDSL=0: consider temperature only
    - IDSL=1: consider temperature and photoperiod
    - IDSL=2: consider temperature, photoperiod and vernalization

  IDSL can be defined at crop variety level. For example, for vernalized winter wheat the default value (in CROP_PARAMETER_VALUE table) is set to 2. For some zones, the value is overwritten to 1 (in VARIETY_PARAMETER_VALUE table).

In the next paragraphs we described the issues solved during 2017 and those ones still open. Closed issues are presented in chronological order, as we solved them. All the patches were promptly sent to Alterra/WER to be applied to their version of Bioma. The dates of the patches represent the date the patch was distributed to Alterra/WER.

1.2.1 Closed issues

1.2.1.1 IDSL parameter was not read correctly

Issue: Bioma did not read correctly the IDSL parameter from database and so vernalization and photoperiod were never applied.

Solution: bug corrected. (Patch 21/02/2017)

1.2.1.2 Modified the query for retrieving crop calendar data by filtering it only on important crops

Issue: when Bioma reads from DB the crop calendar data, it tries to read crop calendar data for every crop in the DB. In case some crops is misconfigured (e.g. grassland, which is not used) Bioma could read erroneous data that cause a crash in the application.

Solution: now Bioma reads only the data of the crops specified in a specific configuration file (called 'CropNamesToAgromanCropNames.txt', already existing and used by the application to convert crop names to crop numbers). Moreover, now Bioma manages properly invalid harvest dates, by setting the harvest date to 31st of December or crop maturity. (Patch 01/03/2017)

1.2.1.3 Differences in photoperiod calculations between Bioma and PCSE

Issue: we spotted a difference in the calculation of biomasses between PCSE and Bioma Wofost. The difference was in the calculation of the photoperiod effect: the PCSE (and CGMS) implementation increases the latitude angle by 4 degrees to take into account the twilight. The angle is needed to take into account that the photoperiodic day length is slightly longer than the astronomic day length because the period of twilight (when the sun is below the horizon) is still counting for the day length experienced by plants.

Solution: Bioma Wofost code was adapted to PCSE code. Now results are identical. (Patch 01/03/2017)

1.2.1.4 Simulation re-initialization after missing weather day

Issue: in case of days with missing weather, Bioma jumps to the next simulation start day (it used to be First of January, now it is dynamical, see below). By doing this, in some cases, it happened that the status of the crop simulation was not re-initialized and so the simulation restarted from the status of the last simulated day. This happened when the restart date was after the sowing date of the next growing season.

Solution: when there is a day with missing weather, now Bioma forces the re-initialization of the status of the simulation. (Patch 10/04/2017)

1.2.1.5 Dynamical simulation restart after a missing weather day

Issue: in case of days with missing weather, Bioma used to jump to the next first of January. This is ok for intra-annual crop, because there is no risk to miss a growing season. This is not acceptable for cross-annual crops because there is the risk to miss a growing season that could be simulated without problems. For example, if the missing weather is during spring or summer, the following winter wheat growing season (which starts in autumn) should be simulated. If the system jumps directly from spring to the next first of January the sowing of that growing season would be skipped.

Solution: when there is a day with missing weather, now Bioma checks for the next day having weather data, and restarts the simulation from that date. (Patch 20/04/2017)

1.2.1.6 Check on not completed growing seasons

Issue: when a growing season does not complete because of some errors (for example because we miss some weather data) Bioma used to save it to the CSV output file and then to the database. So that, in the database there were some growing seasons saved only partially. This could introduce error and biases when calculating long term averages and similar summary statistics.

Solution: now Bioma gives to the user the possibility to save the decadal values of a growing season only when it is completed. By 'growing season' we mean all the decades between sowing and harvest. By 'completed' growing season we mean that the simulation reached the harvest date defined in the crop calendar, so, for example, there was no gap in the weather data. It has no importance whether the crop reached the maturity or not.

The option is defined as a 'configuration item' in the configuration of the persister (PCF file). The item's name is "*Don't save years not reaching harvest day*" and Bioma sets it by default to "*False*", meaning that if the user does not set explicitly the item, the old behaviour is followed.

When the item is set to "*True*", the persister checks if every growing season reached the harvest date. If yes that growing season is saved to the CSV file, otherwise not and there will be a gap in the dates. (Patch 20/04/2017)

1.2.1.7 Problem in crop calendar events in case of leap years, if sowing and harvest are too close

Issue: if between the harvest of a growing season and the planting of the next growing season there is only one day of gap, Bioma does not manage correctly the situation in case of leap years. The result is that Bioma erroneously skips one growing season.

In a non-leap year situation we have, for example,

SEASON	DAY OF PLANTING	DAY OF HARVEST
1997-1998	304	303
1998-1999	304	303

Every year Bioma receives at day 303 an event of harvest, and at day 304 an event of planting (the planting of the next growing season) so everything works correctly.

When there is a leap year, we have

SEASON	DAY OF PLANTING	DAY OF HARVEST
1995-1996	304	304
1996-1997	305	303

Correctly, the two dates of 1996 are 1 day higher. However, during season 1995-1996 Bioma receives at day 304 an event of planting and an event of harvest. Bioma manages first the planting and then the harvest. Then, as a result, Bioma plants and harvests the crop during the same day so season 1995-1996 is not simulated.

This happens for every season preceding leap years.

Solution: we should leave at least 2 days of gap between a harvest day and the next sowing day. We all agreed this is not a problem because in reality the gap is much higher.

1.2.2 Open issues

1.2.2.1 How to manage a possible growing season longer than one year

Issue: one of the options taken into account for initializing the soil water content was to start the simulations up to three months in advance, simulating only bare soil or some filler crop. Unfortunately to extend the simulated growing season by three months causes a problem: there could be an overlap between a growing season and the next growing season. This happens in particular for wheat because there are usually less than three months between an harvest and the next sowing.

The ideal solution is to add a ‘season identifier’ in the output tables (also in the aggregated ones) besides the couple decade/year that we have now. This solution is going to be applied to the operational system at the beginning of the bulletin season 2018.

Partial solution: for 2017, we decided to use the ISW tool to initialize the water content. Therefore, there was no need to run the additional three months.

2 The .NET Core version of the Bioma framework

This chapter describes the changes made to the core of Bioma for running it in the .NET core environment. The .NET Core environment is a free, cross-platform, open source developer platform that Microsoft released during 2016. It is an alternative to the .NET Framework, which is the software framework used to build the Bioma Platform.

To move Bioma from .NET Framework to .NET Core framework has a big advantage: applications built in .NET Core can be run on any operative system (Windows, Linux, Mac) whereas applications built in the .NET Framework are bound to the Windows operative system.

In both the frameworks the programming language is the C#.

It was used version 2.0.2 of the .NET core framework SDK.

The Bioma layers modified are the Bioma Model Layer, the Bioma Composition Layer and the Agromanagement libraries.

For the sake of testing the new code and analysing the impacts, besides of the Core libraries, we converted also a modelling solution. The modelling solution chosen is the Wofost Phenology model. It simulates just the potential phenology of the Wofost model so it is quite simple. Anyway, we ported the whole CropML library to the new environment. CropML contains Wofost, WARM and Cropsyst models and it is composed of circa 60 strategies and 10 domain classes.

At the end of the porting, described in paragraph 2.1, we were able to run the Wofost Phenology modelling solution in the .Net Framework Core.

Later, we decided to make other changes to the Core libraries, to simplify some aspects of the core components and solve problems identified since many years by the Bioma users. These changes are not related to the porting to .NET Core (they could be applied also to the original .Net Framework version) but we identified this moment as the right one to apply them, given the fact that we are already creating a new Core version, not compatible with the original one. These changes are described in paragraph 2.2.

2.1 Porting from .NET Framework to .NET Core

2.1.1 Porting procedure

For the porting from .NET Framework to .NET Core we followed the procedure described on the official Microsoft developers blog:

https://blogs.msdn.microsoft.com/dotnet/2016/02/10/porting-to-net-core/

Here an extract from the blog post:

A ROUGH APPROACH FOR PORTING:

1. IDENTIFY THE PROJECTS THAT YOU WANT TO MOVE TO .NET CORE.
2. UNDERSTAND THE EXTERNAL DEPENDENCIES THESE PROJECTS HAVE AND ENSURE THEY ARE EITHER COMPATIBLE WITH .NET CORE, HAVE EQUIVALENT ALTERNATIVES, OR CAN BE FACTORED OUT.
3. CHANGE THOSE PROJECTS TO TARGET .NET FRAMEWORK 4.6.1. THIS ENSURES THAT YOU CAN USE API ALTERNATIVES WE'VE INTRODUCED FOR CASES WHERE .NET CORE COULDN'T SUPPORT EXISTING APIS. MAKE SURE TO ALSO UPGRADE ANY CONSUMING PROJECTS, OTHERWISE YOU'LL GET COMPILATION ERRORS DUE TO INCONSISTENT .NET FRAMEWORK VERSIONS.
4. RECOMPILE
5. RUN API PORT
6. CHANGE YOUR CODE TO ADDRESS API PORT ISSUES

Here our approach to the procedure:

Point 1) We decided to port the minimum Core for creating a crop simulation modelling solution: CRA.ModelLayer, CRA.CompositionLayer, CRA.Agromanagement, CRA.Agromanagement.Rules, CRA.Agromanagement.Impacts, EC.JRC.MARS.CompositionLayer.Core, EC.JRC.MARS.AgromanagementProviderBaseInterfaces, EC.JRC.MARS.WeatherProviderBaseInterfaces, EC.JRC.MARS.ModelLayer.Data, JRC.IPSC.MARS.Utilities. Besides of the Core libraries, we ported the WofostPhenologyModellingSolution library and its dependencies

Point2) No external libraries to remove. NOTE: in case of porting the Configuration Layer and the CRA.Clima libraries there will be external dependencies to remove and to replace with something else: Ionic.ZIP and NMath.

Point 3) We created a Visual Studio solution containing all the necessary projects. The solution is called "BiomaNetCoreVersion". We modified the target framework from the original one (3.5 for some projects, 4.5 for others) to 4.6.1. It was done without changing any code.

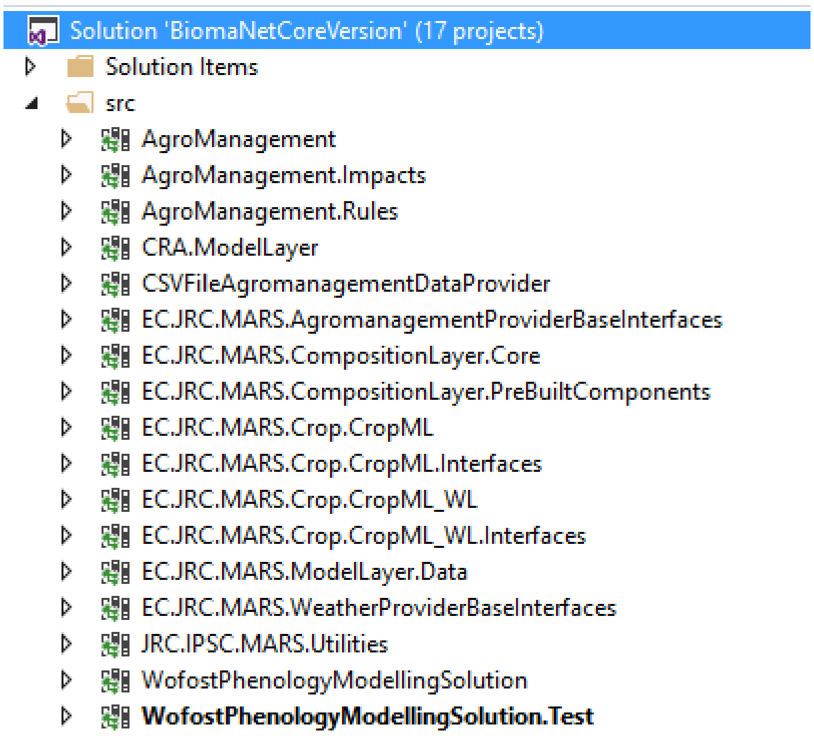

Solution 'BiomaNetCoreVersion' (17 projects)
- Solution Items
- src
  - AgroManagement
  - AgroManagement.Impacts
  - AgroManagement.Rules
  - CRA.ModelLayer
  - CSVFileAgromanagementDataProvider
  - EC.JRC.MARS.AgromanagementProviderBaseInterfaces
  - EC.JRC.MARS.CompositionLayer.Core
  - EC.JRC.MARS.CompositionLayer.PreBuiltComponents
  - EC.JRC.MARS.Crop.CropML
  - EC.JRC.MARS.Crop.CropML.Interfaces
  - EC.JRC.MARS.Crop.CropML_WL
  - EC.JRC.MARS.Crop.CropML_WL.Interfaces
  - EC.JRC.MARS.ModelLayer.Data
  - EC.JRC.MARS.WeatherProviderBaseInterfaces
  - JRC.IPSC.MARS.Utilities
  - WofostPhenologyModellingSolution
  - **WofostPhenologyModellingSolution.Test**

Figure 17- Screenshot of the Visual Studio created

Point 4) Recompilation gave no errors.

Points 5) and 6) are described in the next paragraphs

2.1.2 Running the ApiPort to discover not compatible code

We run the ApiPort utility on a ModelLayer project (EC.JRC.MARS.CropML and all the dependencies). It gave the following feedback

Target type	**Target member**	**Assembly**	**.NET Core App**	**Recommended changes**
T:System.Windows.Forms.MessageBox	T:System.Windows.Forms.MessageBox	CRA.ModelLayer	Not supported	
T:System.Windows.Forms.MessageBox	M:System.Windows.Forms.MessageBox.Show(System.String,System.String)	CRA.ModelLayer	Not supported	
T:System.Windows.Forms.DialogResult	T:System.Windows.Forms.DialogResult	CRA.ModelLayer	Not supported	
T:System.Drawing.Bitmap	T:System.Drawing.Bitmap	CRA.AgroManagement2014	Not supported	
T:System.Drawing.Bitmap	T:System.Drawing.Bitmap	EC.JRC.MARS.Crop.CropML	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].#ctor(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].#ctor(System.Int32,System.Int32)	EC.JRC.MARS.Crop.CropML	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].Address(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].Get(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	

T:System.Double [0:,0:]	M:System.Double[0:,0:][0:,0:].Get(System.Int32,System.Int32)	EC.JRC.MAR S.Crop.Crop ML	Not sup port ed	
T:System.Double [0:,0:]	M:System.Double[0:,0:][0:,0:].Set(System.Int32,System.Int32,System. Double)	CRA.ModelL ayer	Not sup port ed	
T:System.Double [0:,0:]	M:System.Double[0:,0:][0:,0:].Set(System.Int32,System.Int32,System. Double)	EC.JRC.MAR S.Crop.Crop ML	Not sup port ed	

We run the ApiPort utility on the composition layer core project and all its dependencies. It gave the following feedback

Target type	**Target member**	**Assembly**	**.NET Core App**	**Recommended changes**
T:System.Config uration.Configur ationElementColl ection	T:System.Config uration.Configur ationElementColl ection	JRC.IPSC.MARS. Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Config uration.Configur ationElementColl ection	M:System.Confi guration.Configu rationElementCo llection.#ctor	JRC.IPSC.MARS. Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Config uration.Configur ationElementColl ection	M:System.Confi guration.Configu rationElementCo llection.BaseGet (System.Int32)	JRC.IPSC.MARS. Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for

				configuration on type.
T:System.Windows.Forms.MessageBox	T:System.Windows.Forms.MessageBox	CRA.ModelLayer	Not supported	
T:System.Windows.Forms.MessageBox	M:System.Windows.Forms.MessageBox.Show(System.String,System.String)	CRA.ModelLayer	Not supported	
T:System.Configuration.ConfigurationPropertyAttribute	T:System.Configuration.ConfigurationPropertyAttribute	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationPropertyAttribute	M:System.Configuration.ConfigurationPropertyAttribute.#ctor(System.String)	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationCollectionAttribute	T:System.Configuration.ConfigurationCollectionAttribute	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationCollectionAttribute	M:System.Configuration.ConfigurationCollectionAttribute.#ctor(System.Type)	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components,

				expose API for configuration on type.
T:System.Configuration.ConfigurationElement	T:System.Configuration.ConfigurationElement	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationElement	M:System.Configuration.ConfigurationElement.#ctor	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationElement	M:System.Configuration.ConfigurationElement.get_Item(System.String)	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationElement	M:System.Configuration.ConfigurationElement.set_Item(System.String,System.Object)	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationSection	T:System.Configuration.ConfigurationSection	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For

				portable Framework Components, expose API for configuration on type.
T:System.Configuration.ConfigurationSection	M:System.Configuration.ConfigurationSection.#ctor	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Windows.Forms.DialogResult	T:System.Windows.Forms.DialogResult	CRA.ModelLayer	Not supported	
T:System.Configuration.IConfigurationSectionHandler	T:System.Configuration.IConfigurationSectionHandler	JRC.IPSC.MARS.Utilities	Not supported	Use configuration appropriate for your application model. For portable Framework Components, expose API for configuration on type.
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].#ctor(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].Address(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].Get(System.Int32,System.Int32)	CRA.ModelLayer	Not supported	
T:System.Double[0:,0:]	M:System.Double[0:,0:][0:,0:].Set(System.Int32	CRA.ModelLayer	Not supported	

	,System.Int32,System.Double)			

In summary, these are the action points suggested by the ApiPort:

- Everything related to the Windows GUI must be removed (Forms, MessageBox, Drawings, Icons management). The messages to the user must be diverted onto a more generic message system (trace/log)
- The reflection method LoadFrom must be replaced. (In general, some concept in the reflection has changed. More informations can be found at this link: https://blogs.msdn.microsoft.com/dotnet/2012/08/28/evolving-the-reflection-api/)
- The double matrix type (double[,]) is not fully compatible since the methods to set/get a value using the indexes are not supported.
- Classes related to configuration files definitions (system.configuration) must be deleted

2.1.3 Removing not compatible code

These are the actions we applied to the code to solve the problems showed by the ApiPort:

- We removed the classes MapperConfigurationSetting and MapperConfigurationSettings from assembly JRC.IPSC.MARS.Utilities (there will be no mapper in the .NET Core application). The classes gave error because they used System.Configuration
- We replaced the reflection method Assembly.LoadFrom with supported method Assembly.Load
- We replaced MessageBox calls with TraceHelper.TraceEvent calls. Then it is duty of the application to manage properly the messages (e.g display in the console output).
- We removed the usages of System.Drawing.Bitmaps: they were the definition of the libraries icons and logos: icons and logos are should not appear in a Core library
- We removed any reference to the DataSet object because it does not exist in .NET Core
- For bidimensional arrays:
  - We modified instructions like v[j,k] to `(double)v.GetValue(j,k)` where v is a double[,]
  - We changed objects of type double[,] to objects of type double[][]

After applying all these changes, we compiled the code but we were not able to run it: there were too many low level pieces of code not compatible (e.g. read/write file, reflection). So it compiled but there were errors at run time, in particular when running it under the Linux operative system.

2.1.4 Final code modifications

To solve the latest problems we also:

- removed the ImageMap class and all its references (ImageMap was the class used to let ACG application dialog with the Agromanagement core. So it was de fact a piece of graphical interface bounded in the Core.
- removed method for writing XML files (WriteXml) from the Agromanagement core
- removed methods related to cloning of classes

- removed the .NET Framework trace classes. These must be replaced with the Trace mechanism defined in the .Net Core environment.
- replaced ApplicationException exception type to a more generic Exception type

2.2 Refactoring of the code

The refactoring of the code of the Bioma core components described in this chapter is not mandatory for running Bioma in the .NET Core environment but it was applied for improving the components interfaces and behaviour.

2.2.1 Modification to the parameter management

- The parameter class (interface `IParameters`) has been eliminated.
- Methods for setting and retrieving parameters values have been added directly to the strategies, by creating extension methods of the `IStrategy` interface. Through these methods, the consumer can now set the parameters to the strategy one-by-one (by specifying the parameter name and value) or all together by setting a dictionary that contains all the parameters values. The `IStrategy` interface itself is not changed.
- Other methods were added to manage parameter sets directly in the strategy. This functionality replaces the use of 'KeyValues' and 'ParameterSet' objects which were eliminated from the core. These objects were tightly related to ONE of the way parameters could be stored (the XML, MPE format file). Now different sets of parameters can be stored directly into the strategy and then one of the set can be loaded when necessary. (e.g. each set corresponds to a crop parametrization, and then, after an agromanagement event, the correct crop parametrization is loaded). Now this mechanism is completely independent on the way parameters are stored (XMLfiles, database,….). Obviously, users can still use MPE for parametrizing the models and save the parameters to XML format, but this functionality is now out of the Bioma model layer core.
- The `ParametersIO` class was a class designed for two purposes:
  - To inspect through reflection the public properties of a class (e.g. the variables of a domain class)
  - To read/write a parameter class from/to an XML MPE file

  The first functionality was kept, but moved to a class having a more explicit name: `ClassPropertiesHandler`

  The second functionality is no more needed since parameter classes have been eliminated. The `ParametersIO` class has been eliminated too.

- The list of parameters VarInfo, exposed as list of static properties of the strategy classes can be eliminated. The VarInfo of the parameters can now be requested to the strategy. There was a dublication of the same VarInfo values both as static properties and as non static strategy properties. This unnecessary dublication is now eliminated. In case the varInfo list is needed as static context, the new static strategy method `GetParametersVarInfo` can be used (present in every strategy)

  ```
  public static IEnumerable<VarInfo> GetParametersVarInfo()
  ```

- The property

  ```
  IDictionary<string, PropertyInfo> PropertiesDescription { get; }
  ```

  used to be part of `IDomainClass` interface. Now it is part of the new `IPropertiesDescribed` interface. This is to apply the method also to other object than domain classes. Obviously the `IDomainClass` extends the `IPropertiesDescribed` interface.

- The composite design pattern for strategies composition has been improved: now a composite strategy own the instances of the child strategies (before it used to know only the Type of the child strategies). It was *using* the instances of the child strategies, but it had no a direct and organized reference to them.

- The usage of the reflection was limited as much as possible. It is now used only to clone classes (through the interface `IPropertiesDescribed`) and to check conformity of variable types
- Method `SetParametersDefaultValue` of the strategy is now an extension method of `IStrategy` interface and so can be removed from the body of the concrete strategy classes.
- To keep the strategies class more readable, it is possible to split the class code into two files, by using the 'partial' keyword. As a test, we converted two strategies: `Initialization` and `InstantaneousAssimilation`

2.2.2 Modifications to the CRA.AgroManagement component

- The most generic representation of the agromanagement state was an abstract class (`StatesAgroMan`) that had NOT abstract methods or properties. So the abstract class was made coincident with its only implementation (`StatesAgroMan`). `StatesAgroMan` is now the most generic representation of the agromanagement state, and so of course it could be extended if other status variables are needed. In the signatures of the rules methods this class was already used and so non changes are made to the rules. The `CheckRule` method signature keeps to be

  ```
  public bool CheckRule(StatesAgroMan st, IManagement m)
  ```

  NOTE: the fact that the CheckRule interface need an IManagement argument is, in our opinion, meaningless and wrong. We did not change the interface for now but it should be done. The IManagement argument is never used in any implementation.

- The `PhenologicalStates` was moved to an outer scope and now it is defined in its own CS file
- The `CurrentTime` properties (the simulation timing) previously contained static variables. Now the variables are no more static and the whole `CurrentTime` class was added to the agromanagement state (`StatesAgroMan`)
- The `Scheduling` class used to include two different purposes. Now the class has been eliminated and replaced with different classes each one with a single purpose:
  - Load the agromanagement configuration, the rules/impact libraries and read the XML agromanagement file. Now this functionality has been moved to class `AgromanagementFileReader`
  - Perform the time step rule check on the scheduled rules. Now this functionality has been moved to class `AgromanagementController`, which is part of the model runner. Please note that the eliminated Scheduling class did NOT implement the `IScheduling` interface that, without any implementation, was practically useless.

    The workflow for the agromanagement use is now more linear:

    1) The consumer (e.g. an application) uses the `AgromanagementFileReader` for loading the agromanagement libraries and for reading the agromanagement XML file. As an output of this step, the consumer gets the `SchEvents` object containing the scheduled rules/impacts list
    2) At every time step, the consumer passes the `SchEvents` and the agromanagement state (`StatesAgroMan`) to an implementation of `AgromanagementController` (each modelling solution should have its own specific implementation, where all the necessary status variables are set in the specific child of `StatesAgroMan`)

- The prebuilt simulation component AgroManagementSimulationComponent, which was the ready-to-use simulation component to put in a composition layer modelling solution, has been deleted. Now the agromanagement processing is done by the AgromanagementController class, which is a part of the ModelRunner. This means that the agromanagement now is treated no more as one of the components of the simulation but it gained an higher level in the modelling solution structure, being part of the fixed architecture of any modelling solution. This allow to simplify the agromanagement data workflow and to structure better the modelling solutions. Each modelling solution should contain an implementation of the abstract class AgromanagementController. This implementation will fill the agromanagement status with the proper variables, taken from the run time data of the components. So that the AgromanagementController will check the rules versus the specific status of the component. This allows for any developer to extend the StatesAgroMan class, and use any variable as a possible condition for a custom agromanagement rule.
- Simplification in the management classes, since there was a too complex and useless logic abstraction. The IManagementBase interface has been deleted and its methods have been moved into the IManagement interface. The ManagementCollection<IManagement> generic class has been deleted and its methods have been moved to the ActEvents class.

2.2.3 Modifications to the agromanagement data provider interface

The composition layer's IAgromanagementProviderBase has been modified to mirror the changes done in the CRA.AgroManagement component. Before the modification, the interface used to have a method to return the Scheduling object. Now the interface has a method to return the SchEvents object containing the scheduled rules/impacts list. (Location identifier and solar year are passed as parameters because, in some cases, could be that the scheduled events depends on the location/solar year. Otherwise are ignored.)

```
SchEvents GetAgromanagementScheduledEvents(string locationId, int year);
```

This change allow to include in the agromanagement data provider only the logics related to the retrieve of the agromanagement data from a configuration file. A typical agromanagement provider uses the AgromanagementFileReader class to read an agromanagement file and obtain the SchEvents object. The SchEvents object is then used by the simulation component that called the data provider.

In the previous version, the data provider used to return the Scheduling object, which used to contain both the data and the methods to check the rules. The latter is something the data provider should not be aware of.

2.2.4 Modification in the composition layer's simulation components

The ParameterManager class, which was the equivalent at composition layer of the parameter class, was deleted. To set parameters into the simulation component's strategies the consumer can use a set of new methods of the ISimulationComponent interface that mimic the methods for setting the parameters to a single strategy.

The method AreParametersAcceptable of interface ISimulationComponent has been renamed to CheckParameterValues and now it is an extension method

2.2.5 Modifications to the models

The consequences of the changes described in the paragraphs above are minimal on the typical model component. Strategies code is not affected, whereas in the domain classes the references to the `ParametersIO` class must be changed into references to the `ClassPropertiesHandler` class.

In the simulation components must be removed any reference to the deleted `ParametersManager` class and to the parameter classes. At its place, the consumer must use calls to methods `SetStrategyParameterValue`, `AddParameterSet`, `LoadParameters` of interface `ISimulationComponent.`

2.3 Code created

We created three code bases:

1) Solution BiomaCore.sln is the .NET Framework 4.6.1 version, the output of step 4 of the procedure described above in paragraph 2.1.
2) Solution BiomaNetCoreVersionBeforeRefactoring.sln is the .Net Core version obtained at the end of the procedure described in paragraph 2.1, so before the (optional) refactoring described in paragraph 2.2.
3) Solution BiomaNetCoreVersion.sln is the .Net Core version after the refactoring described in paragraph 2.2.

All the three versions of the Wofost phenology modelling solutions give the same simulation results and have the same behaviour.

2.4 Compile and run under Linux

One of the advantages to move to the .Net Core framework is to run the application on different operative system without change any code. We tested the Wofost Phenology console under both Windows 10 and Linux Ubuntu. The behaviour of the application and the simulation results are the same.

Here the procedure for compiling the code to be run under Linux and to run it. The project was built using Visual Studio in a Windows environment.

- In the project.json file of the executable project (in our case project WofostPhenologyModellingSolution.Test) define the runtime appropriate for your linux distribution. Here our example:

  ```
  "runtimes": {
    "win10-x64": {},
    "debian.8-x64": {}
  }
  ```

- Using the DOS command line, go to the folder where the project.json is

  ```
  cd [your path]\NetCoreVersion\Models\WofostPhenology\WofostPhenologyModellingSolution.Test
  ```

- Launch the command

  ```
  dotnet publish -o bin\release\DebianOutput -f netcoreapp1.0 -r debian.8-x64
  ```

  This command will build the project versus the specified framework (-f option) and runtime (-r option) and will copy the output in the specified output directory (-o option).

- Copy in the output directory all the files needed for running the test console (e.g the CSV files containing the input data)
- Copy all the output directory on your Linux machine (we used a docker virtual machine for our tests)

- In your Linux machine, go in the folder where the compiled program is and run and launch the program

  ```
  ./WofostPhenologyModellingSolution.Test
  ```

3 References

Bioma documentation can be found on the Bioma website run by D5 unit at the URL: http://bioma.jrc.ec.europa.eu/documentation.htm

PCSE documentation can be found on the PCSE website run by WER/Alterra at the URL: http://pcse.readthedocs.io/en/stable/

Wofost model documentation (including its water balance) is released in the following user guide: Boogaard, H.L., Van Diepen, C.A., Rötter, R.P., Cabrera, J.M.C.A., Van Laar, H.H., 1998. User's guide for the WOFOST 7.1 crop growth simulation model and WOFOST control center 1.5. Technical Document 52. Winand Staring Centre, Wageningen, the Netherlands, 144 pp.

.NET framework documentation can be found on Microsoft web site, at the URL: https://www.microsoft.com/net/learn/what-is-dotnet

KJ-NA-29019-EN-N

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.

doi:10.2760/713904

ISBN 978-92-79-77324-2