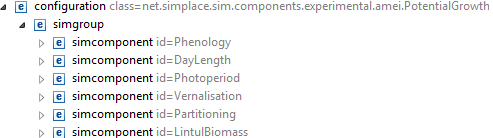
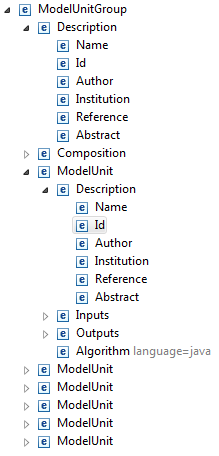
AMEI – Model Exchange Test Case

# Task 1: Import an exported model unit from Bioma into Simplace

The component “Photoperiod” was exported from a strategy of Bioma by Davide Fumagalli into CropML. He provided it to make us test the import into our systems.

First try was creating a module (SimComponent) manually with the default code generator of Simplace. The code generation took about 30 Minutes for the coding and 1 hour for including the component into an existing solution replacing photo response factor in VernalisationAndPhotoresponse component of Lintul2 model. The components were run with the default values of the variables instead of linking them to a configurable input. Testing took quite some time to merge the features accordingly because of small draw backs caused by manual copy paste procedures.

At the end the simulation of the whole crop model was successfully tested producing reasonable output. The structure was as follows:



There is a Simplace Solution documentation together with the data generated and pushed in the GitHub repository to get an overview which data has been used and linked.

## Additional Tasks necessary

Fitting the algorithms part of the C# based code to Simplace was quite simple because the code structure of C# resp. Bioma and Java resp. Simplace is similar. There were no changes on the pasted algorithm necessary. Still I changed the code to fit it to our coding conventions. This took about 3 minutes (for 10 lines of functional code).

One change important has been the including of a factor 60 and conversion of the daylength from hours (Bioma) into minutes (Simplace).

# Task 2: Exporting a model composition from Simplace into CropML

The export of a model composition has the following structure:

Besides the ModelUnits with their meta data, algorithm in a platform specific language and documentation it includes a composition part where inputs from outside the component, outputs to the outside of the component and internal links are explicitly defined.

The export had to be defined to meet fit it to the export features of Simplace and in the same time meet the document type definition of CropML.

At the moment the parametrization (ParameterSet) part and the test cases (TestSet) are still leaking.

The export feature was implemented abstract so that it can be used either for a single Simplace ModelUnit or for a ModelUnitComposition. The Composition was pushed in GitHub repository together with the data and solution information to be able to build an own Simulation Experiment based on the crop model encapsulated in the ModelUnitComposition “PotentialGrowth”.

The development of the export algorithm took about 12 hours in a whole. The export itself takes about one second.

## Additional tasks necessary

### Preparation of the ModelUnits (SimComponents) in Simplace

There are mainly 3 structures of the Simplace code that make it difficult to export them directly to CropML:

1. Use of complex value objects: In this case the InterpolationTable object was used several times in the components that firstly did not let the SimComponent be completely static an on the other hand used included functions in the value objects to retrieve the values in the simulation process. The InterpolationTable objects have been replaced by Doublearrays and the function (FSTFunctions – linear interpolation) was used instead.
2. Use of an init method: Within several components the init initializes the variables. In some there is an additional preparation of static data stored in private (internal) variables that does not have to be updated in the simulation process. These structure has been moved to the process algorithm part by checking, if it has already been initialized and if no it will be done (once).
3. Use of external functions: The called FST-functions are mandatory for the Lintul2 code to keep it readable. There are about 5 functions used regularly in the process code. The functions have to been replaced and have to be included by the importer or the components. The functional class is provided in the task folder in GitHub.

### Discussing and implementing the composition part into the ModelUnitComposition.xml

The implementation of the composition consists of the ModelUnits, the documentation in the description part and the composition links. The links are split into inputs (either inputs form extern or inputs from other internally linked ModelUnits) and outputs (links from component variables to explicit output variables. This is mandatory because in different components there might be output variables with the same name. They have to be merged together or selected, which will be linked to the output of the composition.

### Fitting the output structure to the needs of other platforms

Mainly this task is there to merge the different type information of the variables. Where the one tool uses INTARRAY as datatype another uses Integer[]. Such things have to be mapped by the tools. In detail these type mappings are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable | Tool | CropML | Simplace | OpenAlea | Bioma |
| DataType | **CHAR**  **DATE**  **DOUBLE**  **INT**  **DOUBLEARRAY**  **INTARRAY**  **DOUBLEMATRIX** | CHAR  DATE  DOUBLE  INT  DOUBLEARRAY  INTARRAY  DOUBLEMATRIX |  |  |
| VariableCategory |  | Input  Output  State  Rate  Constant  Privat |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Task 3: Automatic import of Bioma and OpenAlea ModelUnits into Simplace

The import in Simplace can 90% be done with the “SimComponentCodeGenerator”. This has a new method separated from “CSV, XML and DB” named AMEIXML import. The code generator will get its package from the “id” section of the ModelUnit In the XML that makes it necessary to may be adjust this section before running the import.

Main issues are still:

* Information about constants, states , rates are missing – input and output only is used

# Task 4: Automatic import of OpenAlea ModelUnitComposition into Simplace

The import in Simplace will use the Task 3 ModelUnit importer directly. It will first import the ModelUnits and then generate a SimComponentGroup seamlessly.

Main issues are:

* Conversion of the links is not simply copy pasting. It has to be linked to the existing and pre-generated SimComponents. Automation works now.
* Automatic generation of the needed constants and resources will not be possible.

Other possibility could be to import the code into a solution instead of a SimComponentGroup. This may lead to problems with the resource part again.