CropModelMKS: A Non-invasive Crop Model Platform

# Introduction

The CropModelMKS is the abbreviation for the *Crop Model Micro Kernel System*. It is developed aiming at some current popular issues for the next generation crop model.

Crop models have been playing a more and more important role in both the scientific research area and the engineering area since it has been developed and the improvement of them have never stopped since then.

It should be clearly aware of the vast students who are in fact the freshmen in crop modelling but not the professors in this area make up the pillar of the community of crop models. They have little experience in crop modelling, many even have less experience in programming. They should not be expected to design a reasonable structure for his own component.

CropModelMKS is such a crop model platform that is totally begin from the standpoint of these freshmen. We tried our best to reduce barriers for them to get started.

1. There are many legal codes for the crop model, especially written in Fortran.
2. The uneven programming background of the developers
3. The urgence of crafting structure for future challenges

# Features

## Non-Invasive Architecture

A non-invasive structure has many advantages.

### General Structure

There are three levels in the architecture.

The ***Simulator*** is of no doubt the core of the platform. It maintains the life cycle of the whole simulation. It organizes the input parameters and simulated states, execute the simulation and provide the results to clients.

The ***Simulator*** is composed of the ***State***, ***Parameter*** and ***Model\_Module***s. ***State*** is the component that wraps the simulated states, such as the leaf area index, the water content profile, the irrigation amount and so on. And the ***Parameter*** wraps the parameters for a certain library. For example, a ***Parameter*** component contains the CCx, CCo, CGC and so on for the plant part of AquaCrop. The ***Parameter*** is only used for parameterization of a component when initializing. As a result, each ***Library*** matches its only ***Parameter***.

***Model\_Module*** is the actual dispatcher of the crop model components. It maintains the life cycle of a component by judging the running status of the component. If a component finishes its simulation, the ***Model\_Module*** would automatically unload the component then load the next one. There are four ***Model\_Module***s when simulating, and they are identified by the type: climate, plant, environment, and management. The identification has restricted the range of the potential

The ***Library\_Default***, ***Library\_Python***, ***Library\_MATALB*** and ***Library\_Fortran*** are the different loaders of the crop model components developed in different programming languages. Their functions are the same but are realized by different mechanisms. Therefore, they are generalized as the interface ***ILibrary*** for more convenience when called.

The core dispatching the components

The crop model is designed as a non-invasive architecture. The non-invasive architecture means

Create a data pipe might be difficult for the freshmen, so we use the COM to wrap the memory should be shared.

### Data Exchange & Component Communication Mechanism

Component ***State*** and ***Parameter*** are the only two pipelines for data exchange in the simulator for the components of crop models. The interface of ***Parameter*** is constructed one-way only. That is, the component could only inquire the parameters it wanted from the interface. While the interface of ***State*** is duplex. That is, the ***State*** can not only respond to the data request but also receive data from components.

***State*** also plays the role of data bus when data need to be transferred from one component to another.

The communication among the components are crucial to simulate the interactions among the components, especially those between the plant and environment parts. Using the transpiration as an example,

The isolation of each component would prompt the security when replacing a component with another one. To keep such a status of each component, direct communications among them are forbidden. However, the data exchange among components is also of crucial importance, especially when computing the transpiration of plants suffering water stress. In such a scenario, using AquaCrop for example, the transpiration would be calculated in the environment component. When computing, this component needs the data of water sink ability and transpiration ability from the plant component. While the water content profile, field capacity profile and the permanent wilting point profile are need to calculate the shrunk water sink ability under water stress.

The solution should be in such a work flow:

1. The environment component calls the plant component for the sink ability.
2. Upon the request for sink ability, the plant component calls the environment component for the water profile, field capacity and wilting point profile.

To realize the direct communication, each component should keep the objects of all the other three component when running. And should update the

As a result, the State component is also playing the role of the listener for the data request of each component when simulating. Once a component inquire data by name from the State component, it would search the states table it maintaining firstly. If the name of the inquired data is not found in this table, it would search the name in a supports table. If the name exists, it would inquire this data from the respective component. Fig shows the sequence diagram of the process.

### Plug and play(热拔插)

It is normal to consider the plug and play as long as different kinds of components are supported. Moreover, the replacement of the components when running would do simplify some types of simulation, especially for the simulations of rotations and ones for some forecasts.

Inheriting data from the last component is the main obstacle to the replacement. Using the temporary text file does be a method but cost too much time. While under our architecture, it just become a thing never difficult. The later component could just inherit the simulation results easily from the ***State***.

However, the cost of the ability of plug and play leads to the fixed workflow of the Initializing function. It is required that the simulated states should be inquired from the ***State*** before parameterization. If the result of the search is empty, indicating this item has not been simulated, the item should be then searched in the Parameter, and the result should be then sent to the ***State***. Fig shows the detailed workflow.

## Other Features

### Supports for Different Programming Language

Even though almost all the famous crop models were written in Fortran, the new programmers in scientific area now seem to prefer MATLAB or Python.

We have created some simple libraries to test the transfer of the data in this architecture.

C++, C#, Fortran, MATLAB, Python

Some easy components are created to test the inputs and outputs of the interfaces. Even though these components never simulate the crop growth process or the water and nutrient movement, they are the excellent templates that demonstrates how one can access the wanted data from other components. Then the left work for the developer is to fill his components with the formula he would like to use.

### Convenience for the extend analysis

Actually, we used the function Inquiry to allow the clients to get some specific simulation results from the simulator. The simulator itself could be conveniently integrated into the analysis algorithms. However, we still use the class analyzer to wrap the algorithm component and simulator. That is, we not only would like to prompt the reuse efficiency of the components of the crop models but also that of those analysis algorithms.

# Graphic User Interface

# Discussion

We built the assembly with the option “any CPU”. However, this is default for the x86 architecture. This is mainly because that the

We restrict the platform a light solution especially for the freshmen in crop modelling. It is the basic skills to use some programming language to realize the formula used. However, it is always to use the Fortran or master the entire structure are the most obstacles on the way of developing crop models.

We would like the freshmen with the least requirements on programming, namely, who could only use Python or MATLAB to realize formulas he would like to use, could conveniently conduct some improvement on some specific components. This is why this platform created. Beginning with this idea.

However, it is also because of this reason, some assumed

We assume that there would be some usage scenarios such as calling libraries remotely and running the software on different operation system. These requirements are finally deleted that.

## Granularity

Climate, plant, environment and management are the four basic types of components. They, as the smallest granularity is based on the considerations.

The smallest granularity

We do wish the non-invasive architecture could help form a basis for the community of the crop modeler.

Some artificial intelligence algorithms should be running on the

## Parameter Files

Even though the

## Name of the variables

It does not matter how you name your variable in your components, because they are eventually identified by a character string (字符串). However, the name of a variable must accord with the names used in the other components. It requires the developers to follow some standards to normalize the names of their variables. The standard should be but we now just . How we name the variables are also recorded in a XML file, and is stored in the github.io