netherlands eScience center

HyMUSE: a multi-model framework for Hydrology

Inti Pelupessy

EGU General Assembly, Vienna, 08/04/2019

netherlands eScience center

eWaterCycle team

TU Delft

Nick Van de Giesen

Rolf Hut

Jerom Aerts

Netherlands eScience Center

Niels Drost

Ben van Werkhoven

Berend Weel

Gijs van den Oord

Janneke van der Zwaan

Ronald van Haren

Yifat Dzigan

Martine de Vos

Stefan Verhoeven

Inti Pelupessy

Maarten van Meersbergen

+ partners at Deltares, Utrecht University, Wageningen a.o.

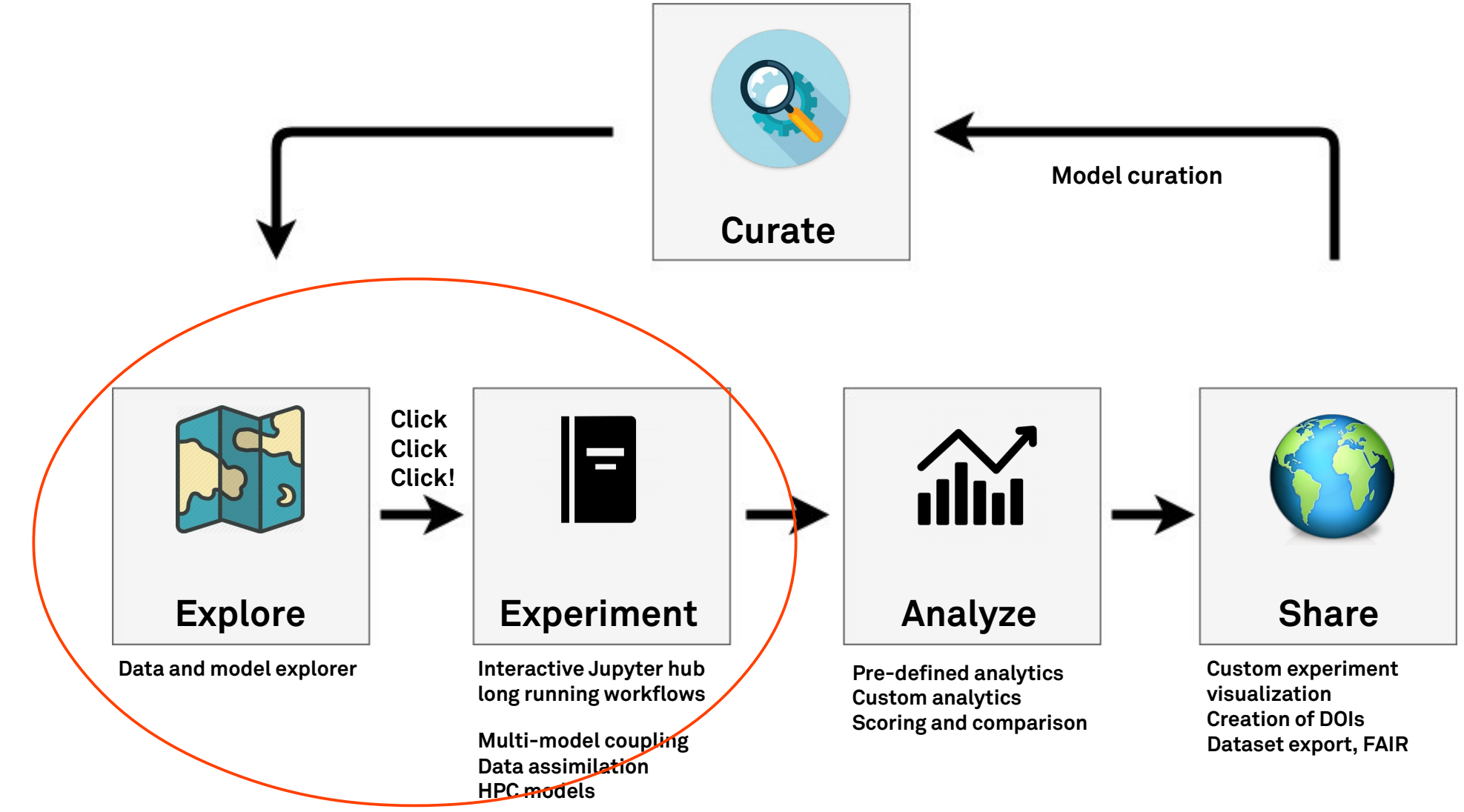
eWaterCycle II Goals

The eWaterCycle II project aims to be a global-scale hydrological modeling platform that allows users to apply hydrological models to answer their research questions.

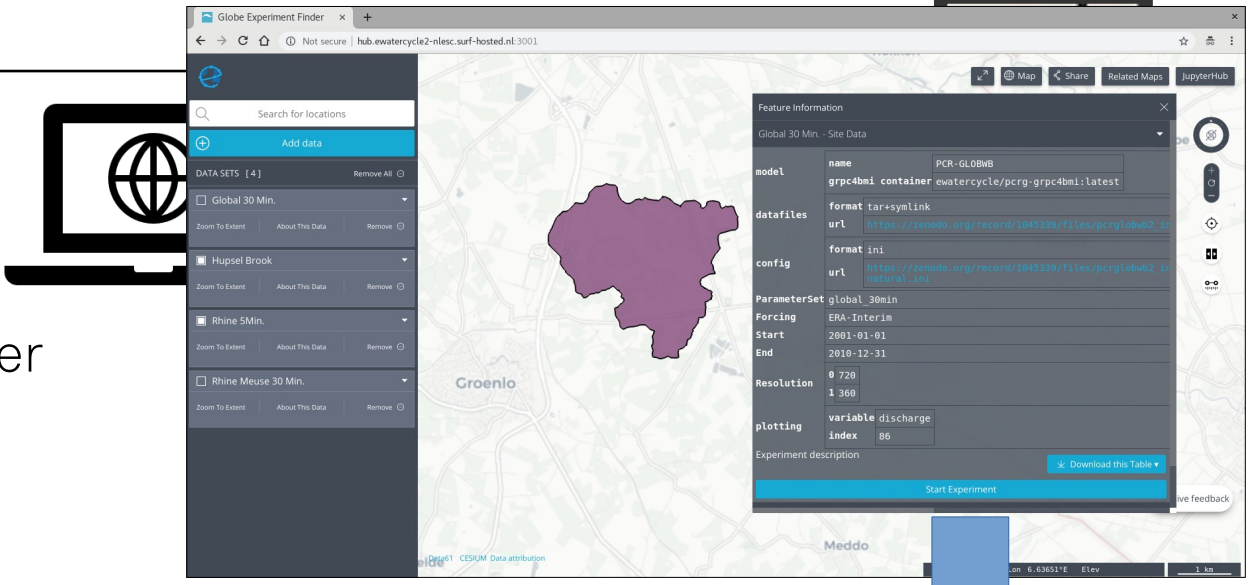
– Combine global hydrological models with local hydrological models, written in different programming languages,
– Recognize findable, accessible, interoperable, and reproducible (FAIR) practices.
– Make it much easier to work together on models.
– Cut down on fte's and time needed to run experiments.
– For this we need a lot of models and a lot of hydrologists to join in.

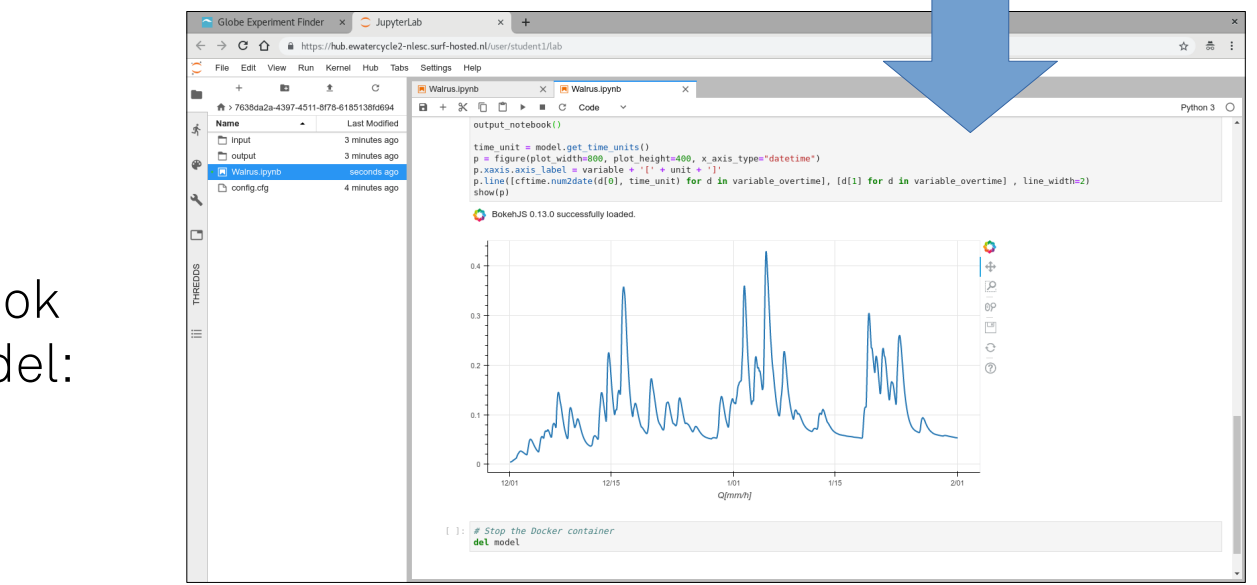
"Towards FAIR & Open Science in Hydrological Modelling"

see also abstracts #12392 (Tue, X3.89) #11014 (Fri, X1.51)
#9276 (Fri 8:30, pico spot 5b) #5580 (Thu, A.17)


Curate
Model curation
Explore
Click
Click
Click!
Experiment
Analyze
Share
Data and model explorer
Interactive Jupyter hub
long running workflows
Multi-model coupling
Data assimilation
HPC models
Pre-defined analytics
Custom analytics
Scoring and comparison
Custom experiment
visualization
Creation of DOIs
Dataset export, FAIR

eWaterCycle II workflow





1. Choose a region in online viewer

2. Get list of available models / forcings / observations for that region.

3. Make a selection and “launch experiment”.

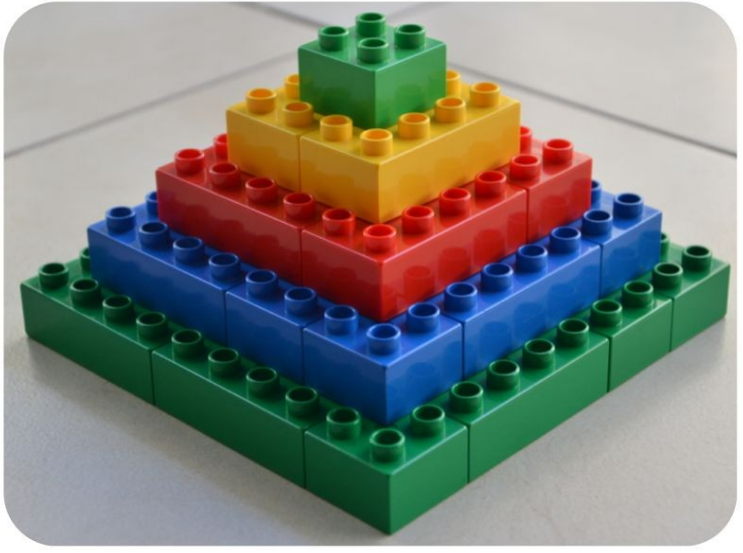
4. Get an (online!) Python notebook with the hello-world for that model: a hydrograph at the basin outlet compared to observations

Building a community

Lorentz workshop in april 2019 to incorporate seven hydrological models

introducing HyMUSE

Problem: we need a method to interact with simulation codes that is:
- simple & intuitive to use
- easy to install
- interfaces codes written in different languages
- uses homogeneous interfaces
- allows the use non-local compute resources



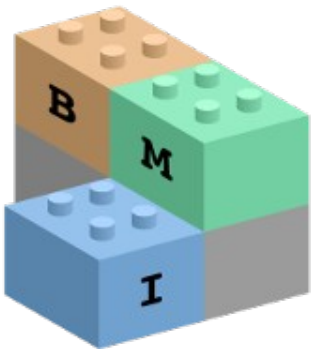
introducing HyMUSE

Problem: we need a method to interact with simulation codes that is:
- simple & intuitive to use
- easy to install
- interfaces codes written in different languages
- uses homogeneous interfaces
- allows the use non-local compute resources

Solution:

AMUSE/ OMUSE

+

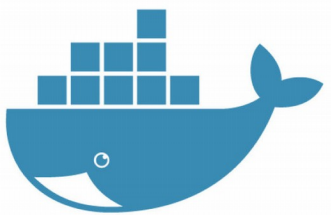


Basic Model Interface

+

GRPC4BMI

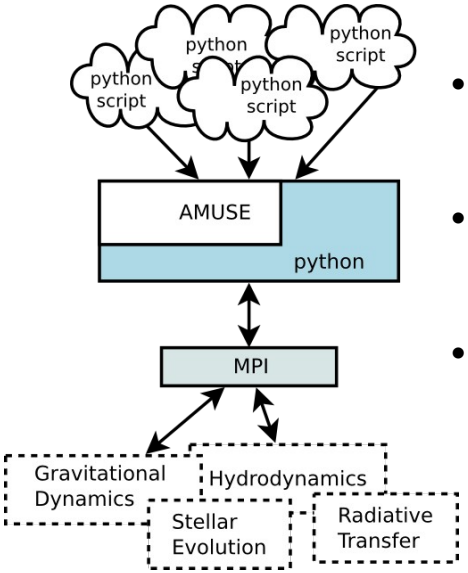
+

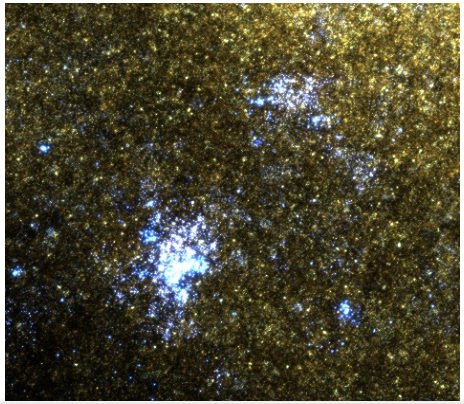


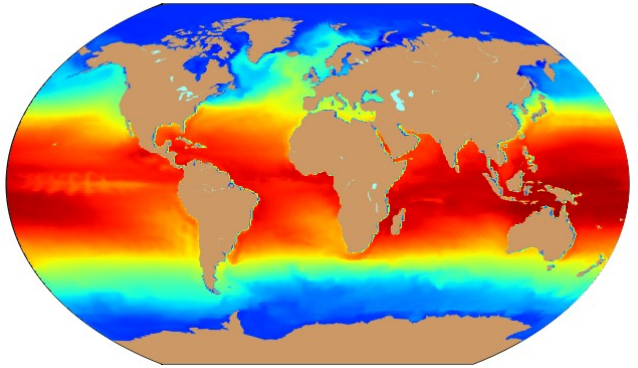
containers

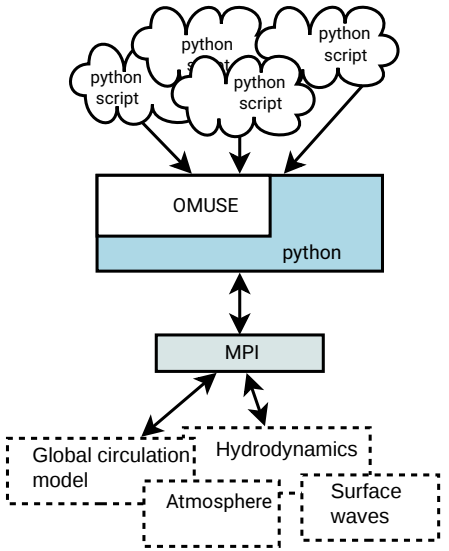
AMUSE & OMUSE

- model coupling frameworks for astrophysics & climate sciences (developed at Leiden Observatory, Utrecht University and the Netherlands eScience Center):
- provide a homogeneous Python enviroment to run community codes
- enable new code couplings and interactions between components
- facilitate multi-physics and multi-scale simulations









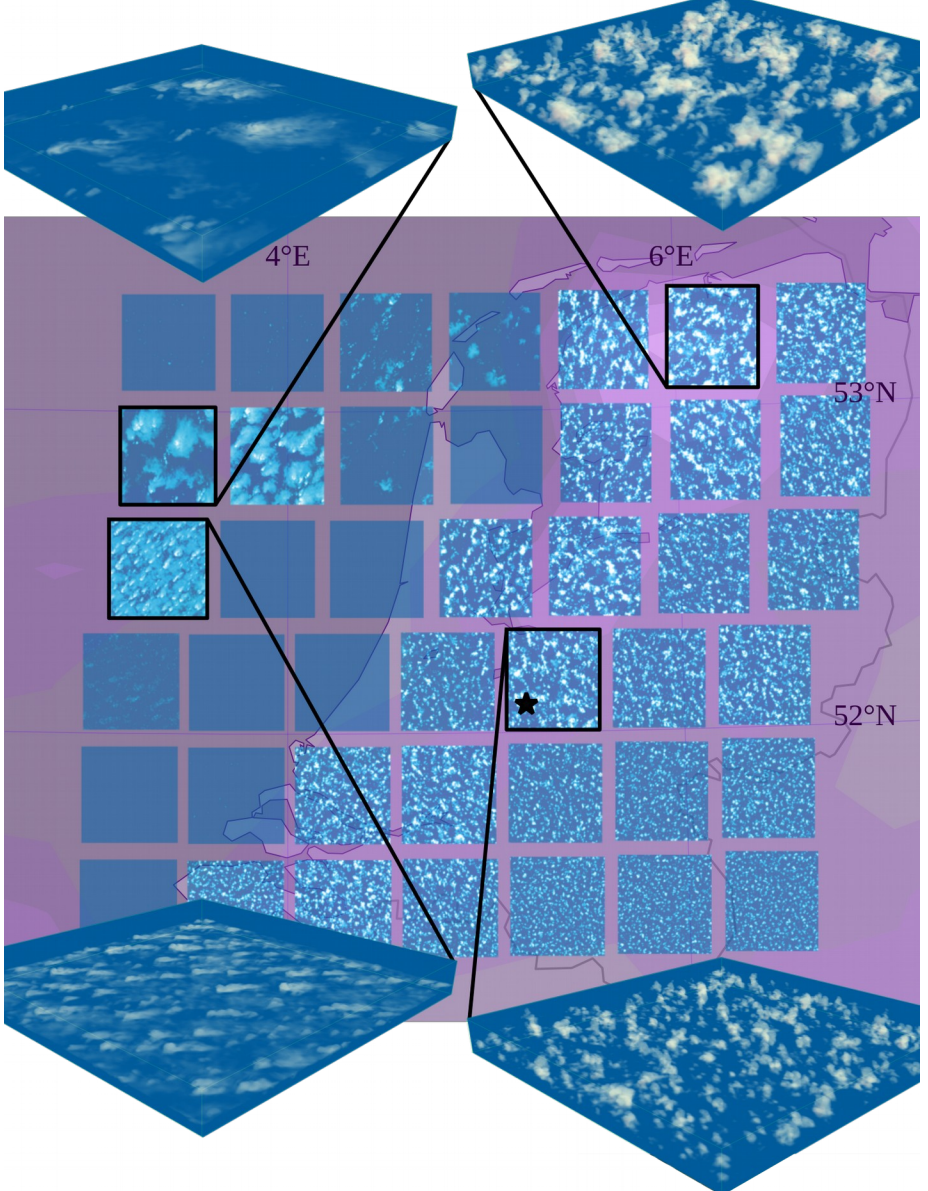
Pelupessy+ 2017
Pelupessy+ 2013

Example:
DALES – OIFS coupling

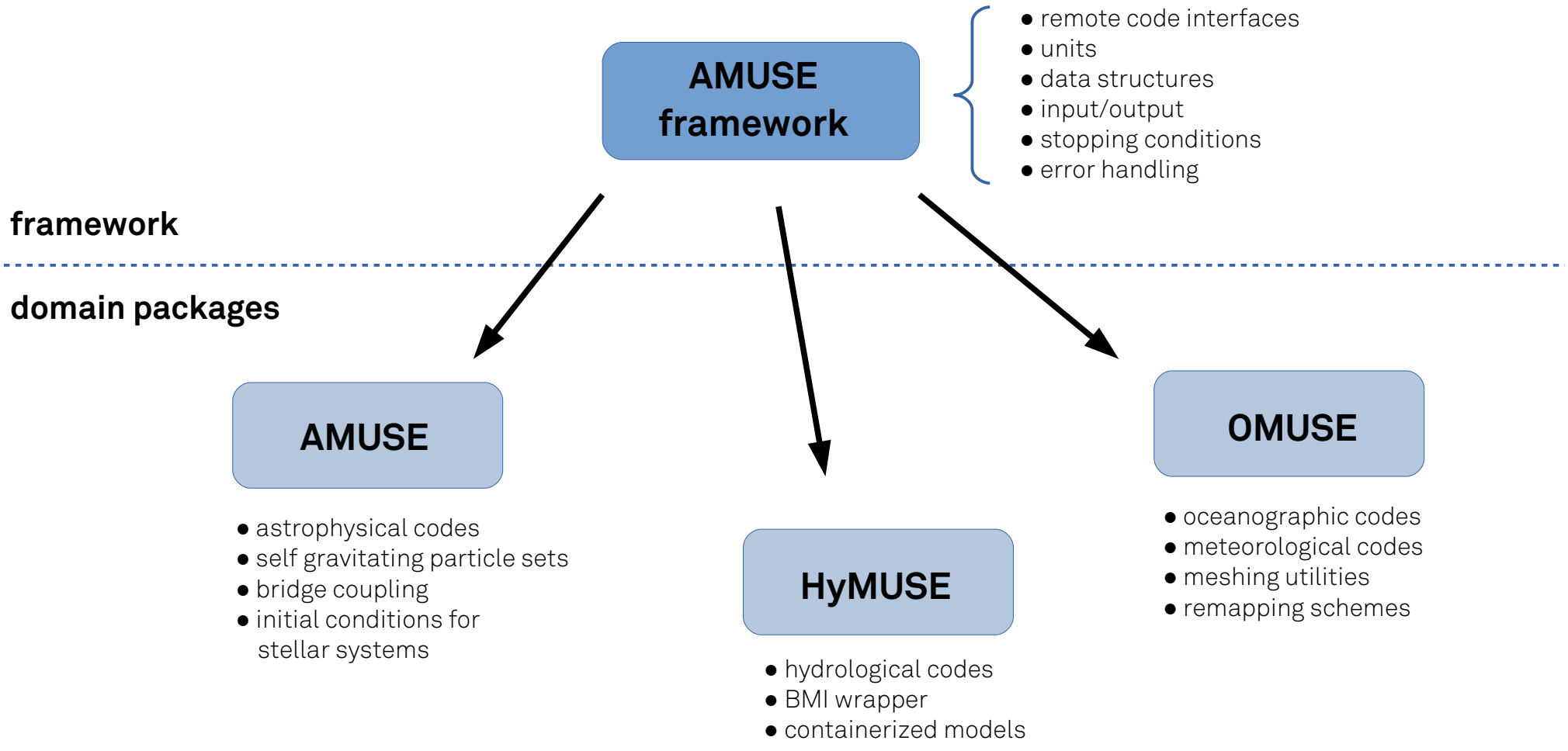
– superparametrization
– embedding of cloud resolving
  in global atmosphere model
– two way coupling
– N models in single global instance

see Jansson et al. #11303 (AS1.1)...

(and other examples:
Pelupessy et al. 2017, GMD 10, 3167)



ABC-MUSE architecture



'Hello Hydro-world'

“imports”

```
from hymuse.units import units
from hymuse.community.pcrglobwb.interface import PCRGlobWB
```

“instantiate code”

```
code=PCRGlobWB()
```

“initialize model”

```
code.parameters.ini_file=”setup.ini”
```

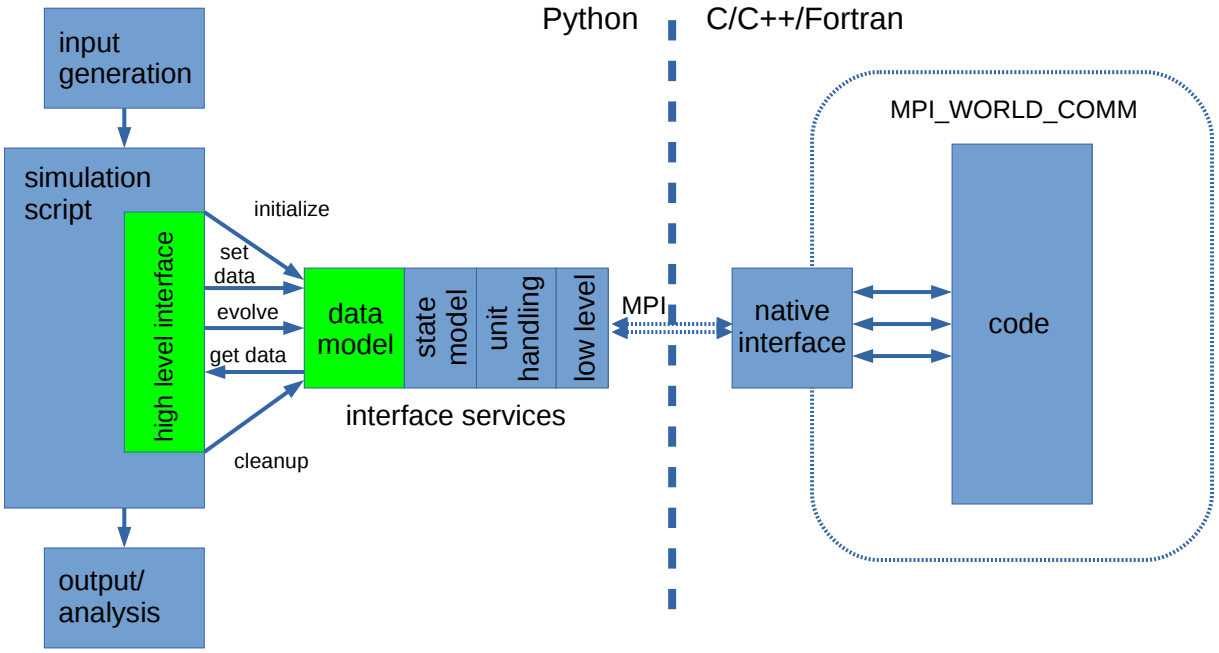
“evolve”

```
code.evolve_model(code.model_time + (1.| units.day) )
```

“analysis”

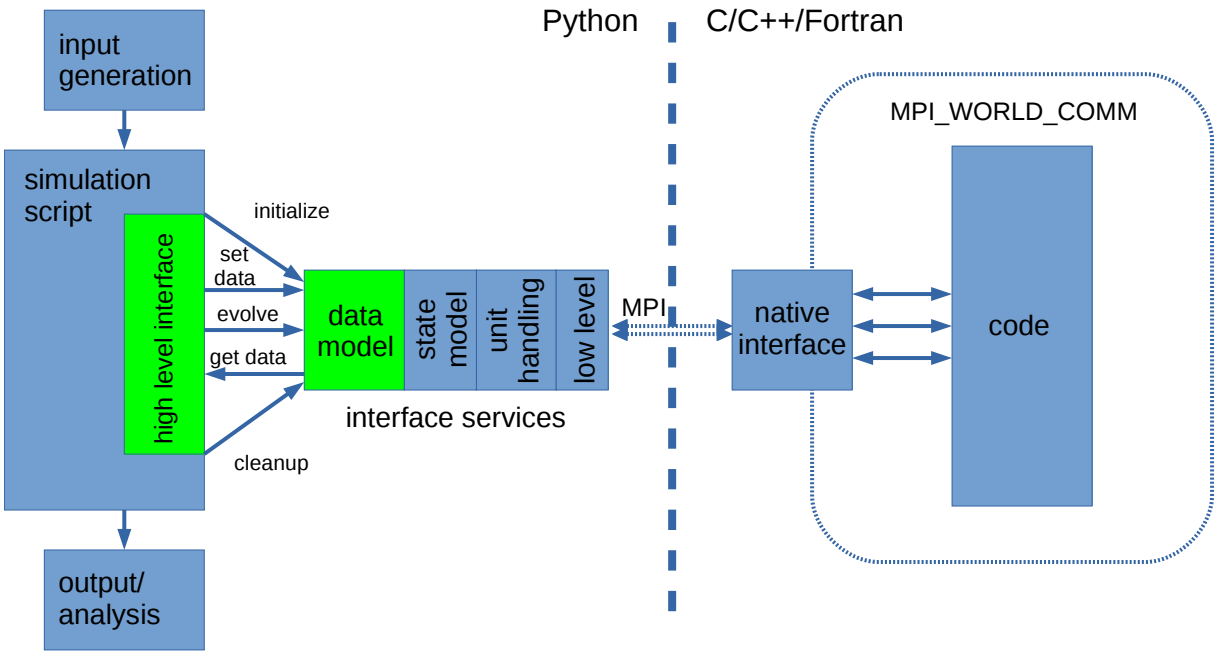
```
station=code.grid_0.samplePoint( lon= 6. | units.deg,
                                 lat= 51. | units.deg )
print("discharge:", station.discharge.in_(units.m**3/units.s) )
```

The HyMUSE interface: a bird's eye view



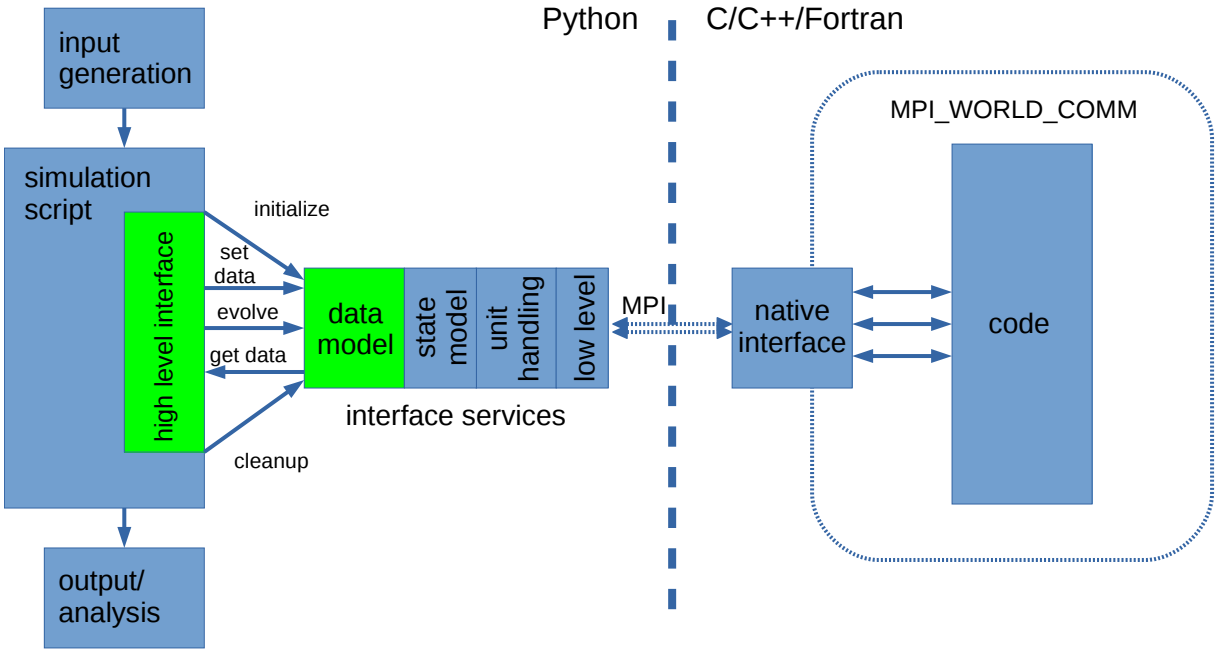
(i) HyMUSE is based on remote code interfaces

The HyMUSE interface: a bird's eye view



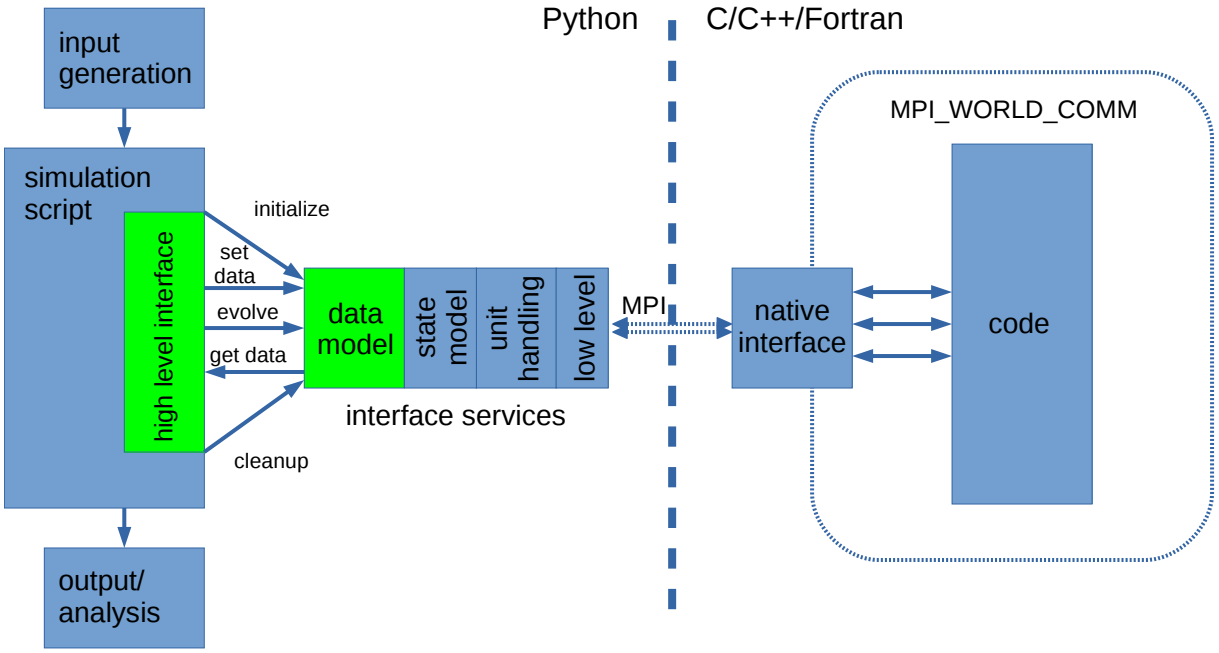
(ii) codes run in parallel & calls can be asynchronous

The HyMUSE interface: a bird's eye view



(iii) HyMUSE provides a number of *interface services:*
unit handling, data structures, code state handling etc..

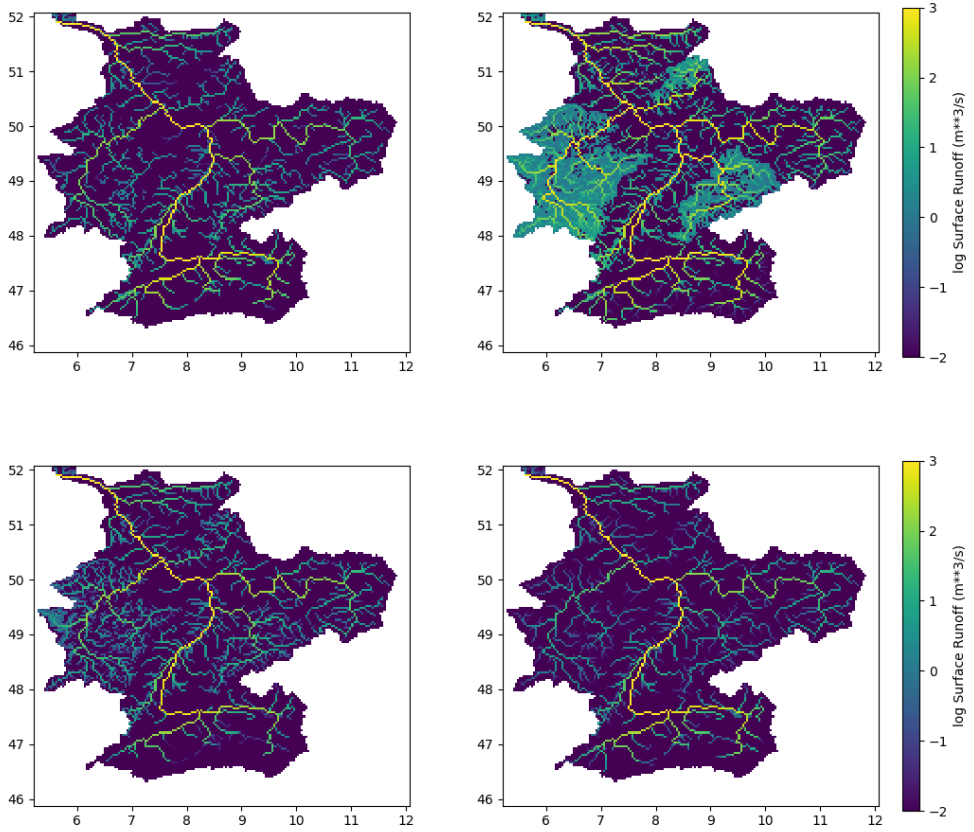
The HyMUSE interface: a bird's eye view



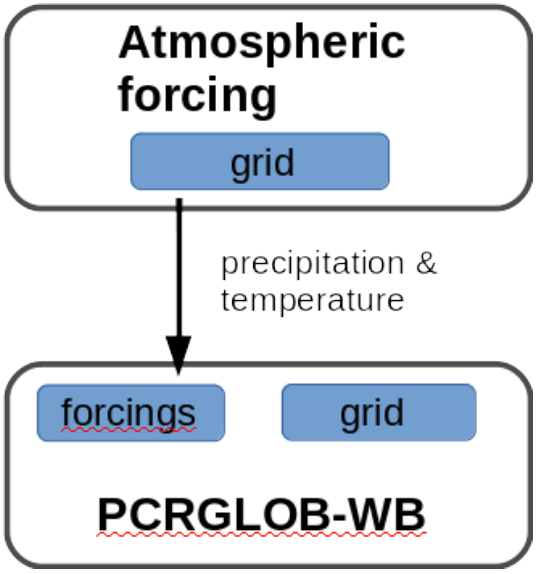
(iv) HyMUSE contributed codes can be containerized and based on BMI, allowing for easy interoperability with e.g. pyMT

What can you do with HyMUSE?

- simplify model setup and runs,
- scripting:
  - event detection,
  - stopping conditions
- 'online' data analysis
- ensemble simulations:
  - parameters searches
  - optimizations (e.g. MCMC)
  - data assimilation
- model comparison: running problems with different codes and methods
- coupling different codes to construct new solvers

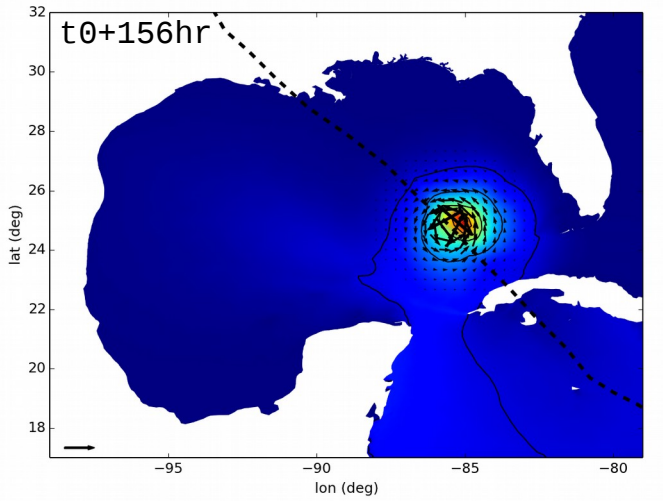


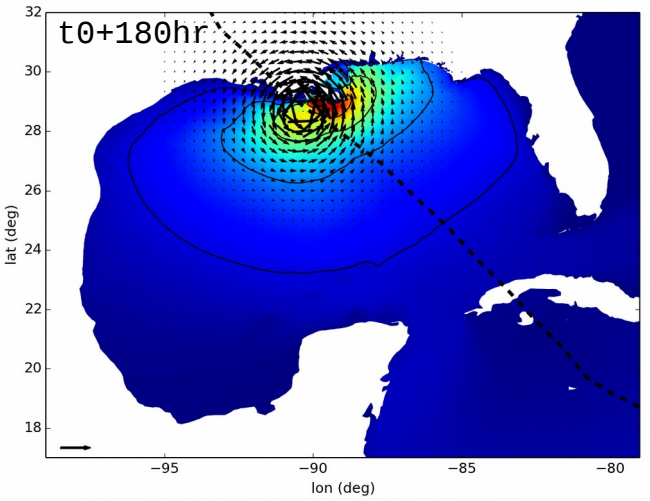
Code couplings with HyMUSE



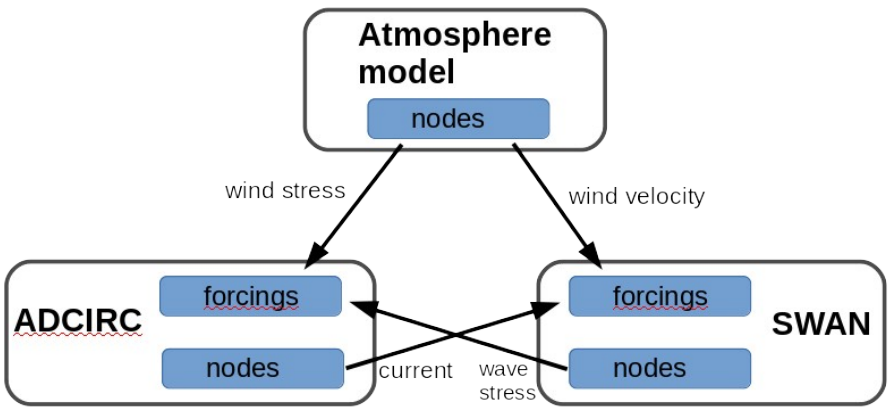

```
pcrglob = PCRGlobWB()

p.parameters.ini_file = ini_file
pcrglob.parameters.use_interface_forcings = True

meteo = netcdf_meteo( "precipitation_2001to2010.nc",
                      "temperature_2001to2010.nc" )

channel = meteo.grid.new_channel_to( pcrglob.forcings )

while pcrglob.model_time<tend:
    meteo.evolve_model( pcrglob.model_time + dt )
    channel.copy_attributes( ["precipitation", "temperature"] )
    pcrglob.evolve_model( pcrglob.model_time + dt )
```

Hurricane Gustav with a coupled hydrodynamic/ wave model





(Pelupessy+ 2017)




```
channel1=hurricane.grid.new_channel_to( swan.forcings )
channel2=hurricane.grid.new_channel_to( adcirc.forcings )
channel3=adcirc.nodes.new_channel_to( swan.forcings )
channel4=swan.nodes.new_channel_to( adcirc.forcings )
while time<tend:
    hurricane.evolve_model(time+dt/2)
    channel1.copy_attributes(["vx","vy"])
    channel2.copy_attributes(["tau_x","tau_y"])
    adcirc.evolve_model(time+dt/2)
    swan.evolve_model(time+dt/2)
    channel3.copy_attributes(["current_vx","current_vy"])
    channel4.copy_attributes(["wave_tau_x","wave_tau_y"])
    time+=dt
```

current status

- basic capability

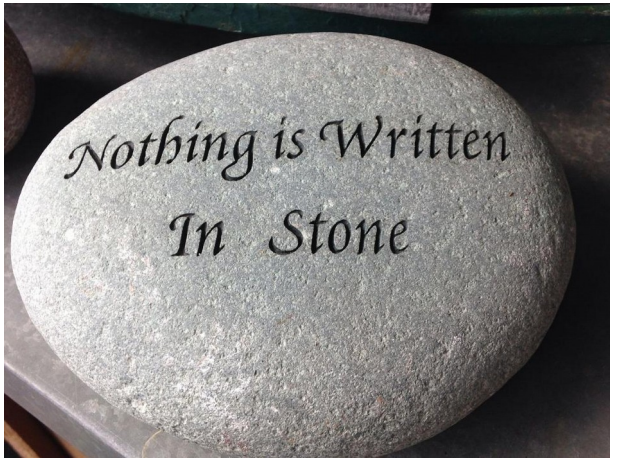
- refinement of high level interfaces needed

- extending community code base

- requirements for "eWaterCycle approved" codes?

- initialization, restarts

- development of BMI:
    * match with eWaterCycle goals?
    * Technical factors: limitations/ shortcomings
    * Human factors: different standards floating around, different opinions on usage



netherlands eScience center

TU Delft

Universiteit Utrecht

Deltares
Enabling Delta Life

WAGENINGEN UR

Conclusions

- eWaterCycle in development as a FAIR platform for hydrology

- HyMUSE aims to be the flexible multi-model framework for the computational core

- It is in development at the Netherlands eScience Center

- Community driven: we *need* your input!

www.ewatercycle.org
lab.ewatercycle.org
github.com/ewatercycle/hymuse

netherlands eScience center

TU Delft

Universiteit Utrecht

Deltares
Enabling Delta Life

WAGENINGEN UR

Thank you!