natESM

DKRZ DEUTSCHES KLIMARECHENZENTRUM

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

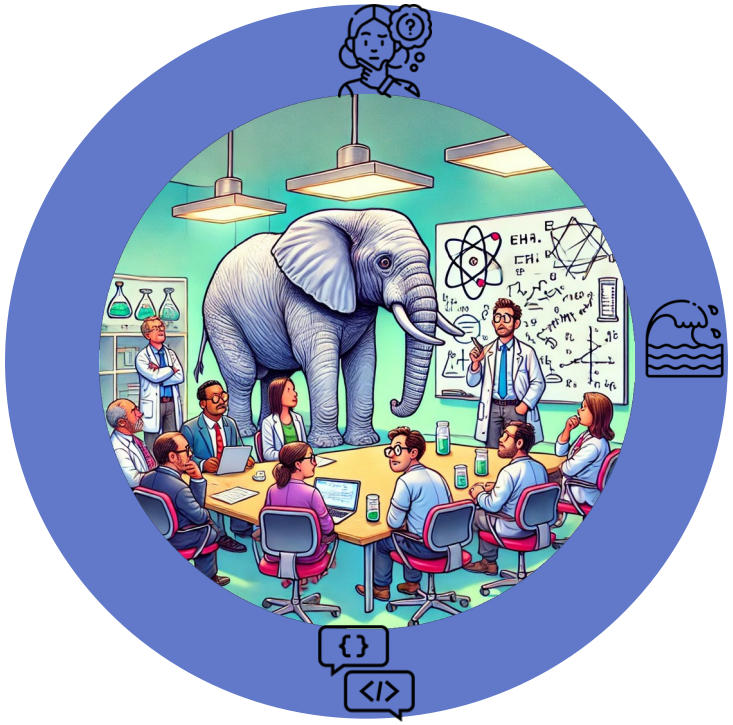
DKK Deutsches Klima Konsortium

Federal Ministry of Education and Research

Community Workshop 2025

YAC, ComIn, and beyond – what do you need from natESM?

Breakout Group: YAC, ComIn and Beyond



Mahnoosh Haghighatnasab (DWD) presented the basic idea and the current development status of the Community Interface (ComIn). She explained the use of the embedded Python interpreter and how ICON fields can be modified.

- The talk triggered many follow-up questions about concurrent execution of plugins, using different grids and domain decompositions.

Moritz Hanke (DKRZ) gave a short introduction on ICON"s coupler library YAC.

- Among the various presented use cases, the application of YAC to model input and output was given particular attention.

- Besides, YAC's built-in functionality to write and read interpolation weight files was discussed in the context of offline interpolation tasks, as well as its dependency on the YAXT communication library.

Breakout Group: YAC, ComIn and Beyond

Bastian Kern (DLR) summarized the design concept and the development history of the Modular Earth Submodel System MESSy.

- Here ICON constitutes only one of the regional and global NWP and climate models than MESSy can connect to. This interface was recently reimplemented as a ComIn plugin with the help of a natESM sprint.

- The talk also pointed out certain *restrictions*: What could be implemented in a MESSy-ComIn plugin - and which MESSy features require an adaptation of the ICON model itself. For example, temporary variables would have to be turned into global fields to become accessible by the plugins.

- The presentation concluded with a recipe on how to adapt an existing software package to ComIn in general.

Breakout Group: YAC, ComIn and Beyond

Afterwards, a **general discussion** followed, focusing on guidelines and the pros and cons of the different interface approaches for specific purposes.

Applications discussed: CLEO, ensemble data assimilation, ML experiments, the LSM sprint… reasons why YAC, ComIn, or a direct integration into the ICON code was chosen as an approach.

Criteria / Guidelines:

- Choice of programming language: Is Python or C++ better suitable for the task?
- Ease of access, eg. for master students
- Objective: not to enter the main code of ICON?
  Complexity in doing a MR to the big ICON source code
- Need for a stable interface for external code contributions, compatibility with future ICON versions

Additional strengths of the YAC coupler:

- does the model component need to run on different grids?
- placement on heterogeneous hardware?

Breakout Group: YAC, ComIn and Beyond

Thinking the vision of external plugins consequently to the end, would it make sense if ICON became a “skeleton”?

Certainly not.

- Switching off parameterization schemes involves tuning of the model as well, which takes time. The tuning of model components relies on a stable set of integral parameterizations.

- "Open development" would be pointless if the external contributions would not be fed feedback into ICON at some point. At least, certain plugins should be shipped with ICON.

Finally, the definition of "core model components" often is a rather political question.

Breakout Group: YAC, ComIn and Beyond

How could natESM address the above concerns and requirements?

- Can natESM offer a kind of platform or website to exchange ComIn plugins?

- Refactoring of ICON model needed in some parts:
  Eg., switching off parameterization schemes (for replacement) still not easily done in ICON.

- Could YAC be extended to a true 3D interpolation algorithm?

Uncertainty about the stability of interfaces with the background
Future WarmWorld plans: Single precision, refactoring the code into C++?