ESM-Tools - A modular approach of an Earth-System-Model infrastructure software

Nadine Wieters, Sebastian Wahl, Miguel Andrés-Martínez, Paul Gierz, Jan Streffing

HELMHOLTZ

March 7, 2024

Table of Contents

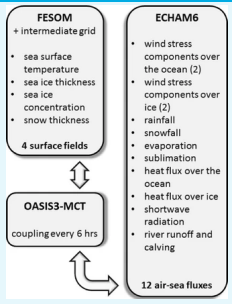
Motivation: Enable a modular and configurable ESM setup

- ESM-Tools (Barbi et al., 2019): A **workflow tool** to **build Earth System Models** (ESM) and **run climate simulations** on **different High Performance Computer** (HPC)
- Provide an easy-to-use tool for ESM researchers:
  - Climate researchers with or without deep knowledge about programming and/or software engineering practices to have a low level entry in climate simulations
  - Experienced scientists who use different models on different HPCs to make daily work more comfortable
- Modular software
  - improves code development
  - enable extendability
  - enable interoperability
  - enhance maintainability

ESM

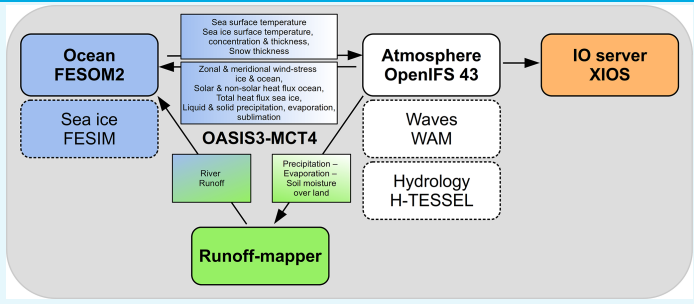
- modular workflow and infrastructure software for Earth System Modelling (ESM)

AWI-CM 1



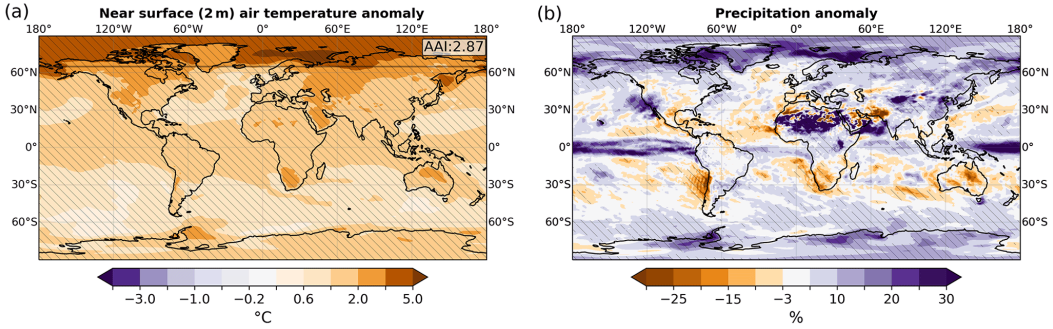
Sidorenko et al. (2015)

AWI-CM 3



Streffing et al. (2022)

AWI-CM3



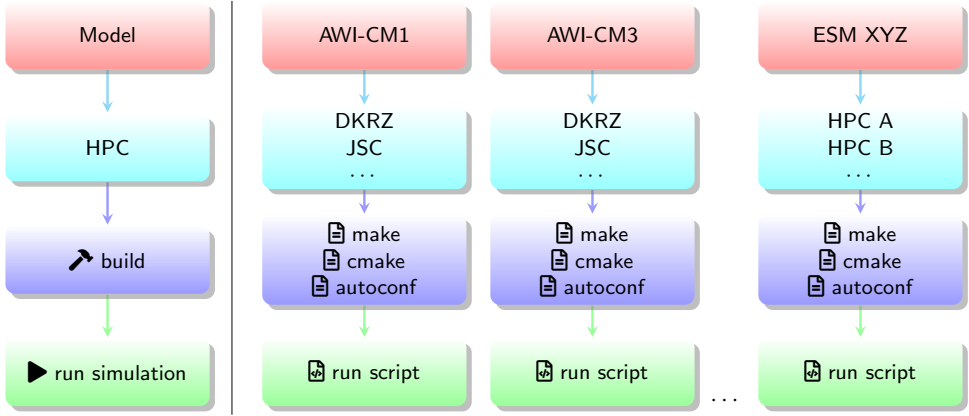
AWI-CM3 temperature (a) and precipitation (b) anomaly (Streffing et al., 2022)

ESM workflow

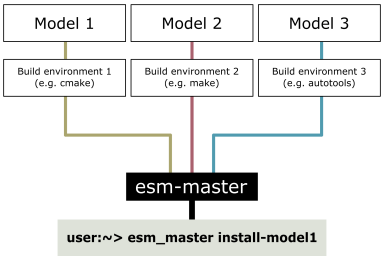
From user/researcher perspective

- **Simulation** depends on research subject, field of domain, → choose model
- **model** code availability, source code, input data, forcing data → choose HPC resources
- **HPC** software libraries to build model, hardware to run simulation, accounting of CPU hours

ESM infrastructure



ESM-Tools user commands



Tasks:
get-, comp-, update-, clean-, status-, log-, install-, recomp-

Barbi et al. (2019)

esm_master

- download/clone model code
- build model executable(s)

```
esm_master install-awicm1
```

esm_runscripts

- run simulation

```
esm_runscripts run.yaml -e expid -c -t task
```

Software demands

- address **different models and HPC** requirements
- **easily expandable** in order to include future ESMs and HPCs
- expandability to the functionality of the software should be **customizable by the user/researcher** of the software
- expand the functionality by **not changing the back-end source code**

Modularity

- code development
- software usage

Software design choices

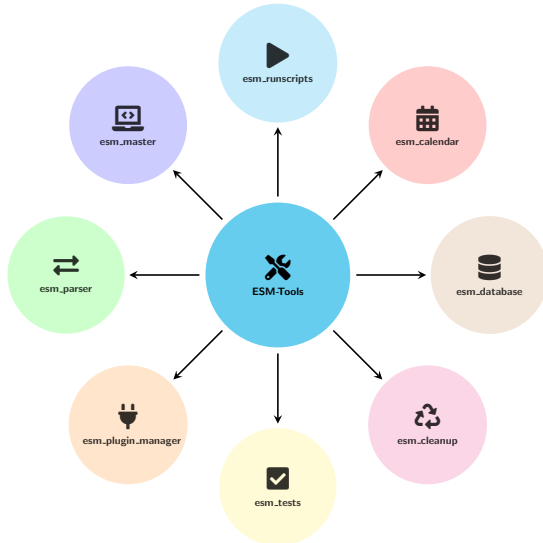
Code development:

- separation of concerns:
  - separate code and configuration: HPC- and model-agnostic **Python back-end**, **YAML configuration files**
  - modular configuration files

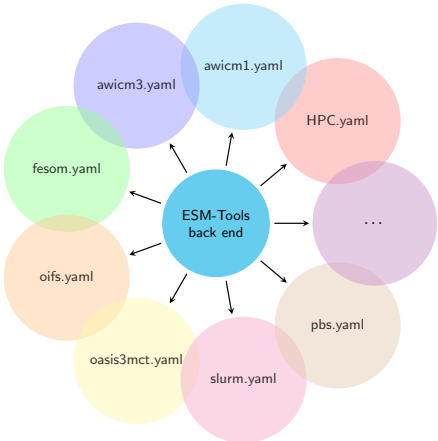
Software usage:

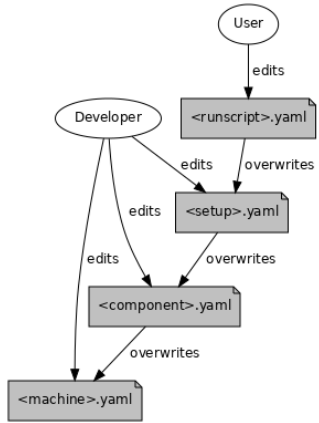
- enable an extended functionality to the modular and hierarchical configuration files by applying a special **configuration file syntax** (esm_parser),
- provide an **adaptable workflow** and **plug-in manager** that is configurable by the advanced user to extend and add new functionality

Core modules



Modular and hierarchical configuration





Barbi et al. (2019)

Configuration functionality (esm_parser)

- creating and accessing variables from different sections/config files

```
ini_restart_dir: "${general.ini_restart_dir}/fesom/"
```

- math and calendar operations

```
runtime: $(( ${end_date} - ${time_step}seconds ))
```

- adding and removing elements from lists and dictionaries

```
list1:
    - element1
    - element2
add_list1:
    - element3
    - element4
```

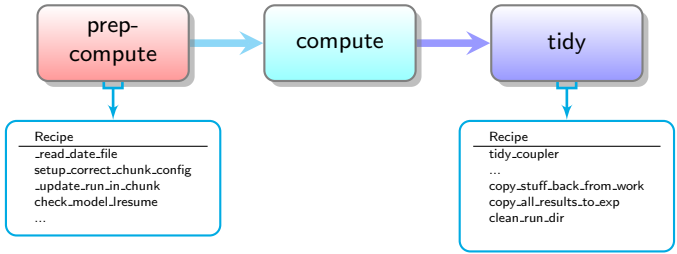
- changing Fortran namelists

```
namelist_changes:
    namelist.echam:
        runctl:
            out_expname: ${general.expid}
            dt_start:
                - ${start_date!year}
                - ${start_date!month}
```

- implement select-case/switch statements via choose-blocks

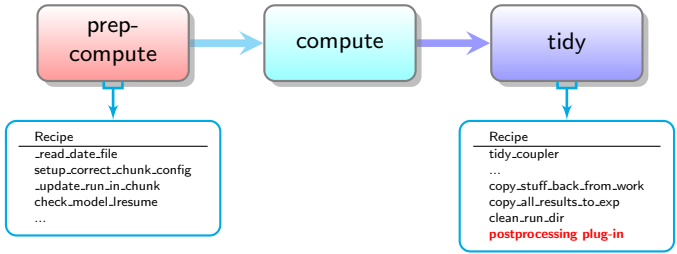
```
resolution: CORE2
choose_resolution:
    CORE2:
        nx: 126858
        mesh_dir: "${pool_dir}/meshes/"
        nproc: 288
        time_step: 450
    GLOB:
        nx: 830305
```

Workflow and plug-in manager



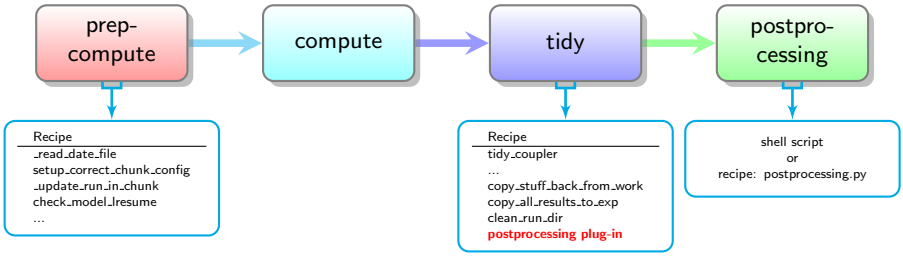
- workflow phases are python modules
- workflow phase steps are defined as recipes in workflow configuration
- add new functionality as plug-in into a recipe: source code python function, python function in other repository

Workflow and plug-in manager



- workflow phases are python modules
- workflow phase steps are defined as recipes in workflow configuration
- add new functionality as plug-in into a recipe: source code python function, python function in other repository

Workflow and plug-in manager



- add new functionality as plug-in into a recipe: source code python function, python function in other repository
- add new workflow phase: shell script, source code python function
- complete recipe is customisable by user

Advantages

- Splitting configuration and code enables a good **maintainable code development**
- The most important **changes** in the user **PRs** are more and **more related to the configuration files and sample runscripts** (at least of what we know about)
- Very little changes on back-end code
- Less need to update core code except for bugfixes
- Reduces support, since a lot of changes can be done in configuration
- Modular configuration files enable the **reusability of configuration** across modelling groups, e.g. for model components

Challenges, lessons learned, solutions

- In order to take advantage of the expandability options:
  - the special syntax of configuration files need to be learned by user
  - the implementation of new functions need to be clear and comprehensible to user
- In order to provide a stable functionality of ESM-Tools
  - changes to configuration and workflow needs a good testing strategy

Challenges, lessons learned, solutions

- provide **documentation** and examples of the configuration syntax
- documentation and practical **use case examples** for workflow manager
- **syntax check** for configuration files
- establish best practices in applying changes to (default) configurations, e.g. templates
- possible errors demands a **good and speaking error message**s and good coverage of exception handling and a good availability of **logging information**
- **unit tests** for core functionality
- good coverage of **integration tests**
- establish a well used user forum for help and discussion on issues and problems
- well established **user communications** on changes (bugfixes, changes to config, mayor releases, ...)

Outlook

- enable and transfer the software into an open source community software project
- get a stable core code
- minimal to support and bug fixes

General information

- https://esm-tools.github.io
- https://github.com/esm-tools/esm_tools
- https://esm-tools.readthedocs.io
- 10.5281/zenodo.3737927
- https://github.com/esm-tools/workshops/tree/main/202207_AWI_ESM-Tools
- ESMTools: Workshop material
- Monthly online user meeting
- User support
  - esm-tools-info@listserv.dfn.de
  - Github Issues, Discussions
- Mailing list https://www.listserv.dfn.de/sympa/subscribe/esm-tools-users

Nadine Wieters
nadine.wieters@awi.de

Thank you for your attention!

References I

Barbi, D., Wieters, N., Cristini, L., and Gierz, P. (2019). Esm-tools: A common infrastructure for modular coupled earth system modelling. *GMD*.

Sidorenko, D., Rackow, T., Jung, T., Semmler, T., Barbi, D., Danilov, S., Dethloff, K., Dorn, W., Fieg, K., Goessling, H. F., Handorf, D., Harig, S., Hiller, W., Juricke, S., Losch, M., Schröter, J., Sein, D. V., and Wang, Q. (2015). Towards multi-resolution global climate modeling with echam6–fesom. part i: model formulation and mean climate. *Climate Dynamics*, 44(3):757–780.

Streffing, J., Sidorenko, D., Semmler, T., Zampieri, L., Scholz, P., Andrés-Martínez, M., Koldunov, N., Rackow, T., Kjellsson, J., Goessling, H., Athanase, M., Wang, Q., Hegewald, J., Sein, D. V., Mu, L., Fladrich, U., Barbi, D., Gierz, P., Danilov, S., Juricke, S., Lohmann, G., and Jung, T. (2022). Awi-cm3 coupled climate model: description and evaluation experiments for a prototype post-cmip6 model. *Geoscientific Model Development*, 15(16):6399–6427.