Geosci. Model Dev., 14, 4051–4067, 2021
https://doi.org/10.5194/gmd-14-4051-2021

Geoscientific Model Development
Open Access
EGU

ESM-Tools version 5.0: a modular infrastructure for stand-alone and coupled Earth system modelling (ESM)

Dirk Barbi[1], Nadine Wieters[1], Paul Gierz[1], Miguel Andrés-Martínez[1], Deniz Ural[1], Fatemeh Chegini[1,3], Sara Khosravi[2], and Luisa Cristini[1]

[1]Alfred Wegener Institute Helmholtz Center for Polar and Marine Research, Bremerhaven, Germany
[2]Alfred Wegener Institute Helmholtz Center for Polar and Marine Research, Potsdam, Germany
[3]Max Planck Institute for Meteorology, Hamburg, Germany

Correspondence: Dirk Barbi (dirk.barbi@awi.de)

Received: 8 April 2020 – Discussion started: 11 August 2020
Revised: 16 March 2021 – Accepted: 3 June 2021 – Published: 30 June 2021

Abstract. Earth system and climate modelling involves the simulation of processes on a wide range of scales and within and across various compartments of the Earth system. In practice, component models are often developed independently by different research groups, adapted by others to their special interests and then combined using a dedicated coupling software. This procedure not only leads to a strongly growing number of available versions of model components and coupled setups but also to model- and high-performance computing (HPC)-system-dependent ways of obtaining, configuring, building and operating them. Therefore, implementing these Earth system models (ESMs) can be challenging and extremely time consuming, especially for less experienced modellers or scientists aiming to use different ESMs as in the case of intercomparison projects.

To assist researchers and modellers by reducing avoidable complexity, we developed the ESM-Tools software, which provides a standard way for downloading, configuring, compiling, running and monitoring different models on a variety of HPC systems. It should be noted that ESM-Tools is not a coupling software itself but a workflow and infrastructure management tool to provide access to increase usability of already existing components and coupled setups. As coupled ESMs are technically the more challenging tasks, we will focus on coupled setups, always implying that stand-alone models can benefit in the same way.

With ESM-Tools, the user is only required to provide a short script consisting of only the experiment-specific definitions, while the software executes all the phases of a simulation in the correct order. The software, which is well documented and easy to install and use, currently supports four ocean models, three atmosphere models, two biogeochemistry models, an ice sheet model, an isostatic adjustment model, a hydrology model and a land-surface model. Compared to previous versions, ESM-Tools has lately been entirely recoded in a high-level programming language (Python) and provides researchers with an even more user-friendly interface for Earth system modelling. ESM-Tools was developed within the framework of the Advanced Earth System Model Capacity project, supported by the Helmholtz Association.

1 Introduction

Earth system models (ESMs) are widely used for studying past, present and future climates, processes in the different Earth system compartments (e.g. atmosphere and ocean), as well as the interactions between them. Therefore, ESMs can include several model components (e.g. atmosphere, ocean, ocean biogeochemistry, land and sea ice, land biosphere, hydrology) which are developed independently by different research groups. Various ESMs have been developed in recent years by using different model components and couplers. Therefore, in order to be able to apply these models, the users need to acquire knowledge of various model- and system-dependent parameters. This often proves time consuming and challenging for scientists who are new to numerical modelling or are approaching a numerical model they are not familiar with. ESMs often lack modularity as the primary

focus during development is on the scientific question to be studied rather than technical standards or reusability of the code. This lack of modularity in coupled systems can make it very inefficient when it comes to replacing certain model components. In the context of further high-performance computing (HPC) development, e.g. the full use of emerging heterogeneous HPC systems, the use of coupled model systems becomes particularly complex (see also Bauer et al., 2015, Schulthess et al., 2019). This issue can be further enhanced by lack of training in software development, often (but not always) amongst early career scientists, making the procedure non-trivial and error prone. To address these issues, we developed ESM-Tools – a software that considerably reduces the difficulty of applying ESMs by providing a modular external modelling infrastructure that allows scientists to work with stand-alone as well as coupled setups in a very intuitive and straightforward manner.

The complete workflow for ESM applications includes tasks such as obtaining and compiling all source codes, managing input data and configuration files for model setup, submitting the executable to a multi-processor system, monitoring and logging the process and managing/post-processing output data. These tasks require knowledge of the parameters used by the ESM such as build environment, configuration files (e.g. Fortran namelists) and I/O structure as well as knowledge of the specific HPC environment (e.g. installed libraries and batch system). This often results in an ESM modeller having to deal with numerous technical issues. The ESM-Tools software provides standard solutions to typical tasks occurring within the workflow of Earth system modelling, such as calendar operations, data post-processing and monitoring, sanity checks, sorting and archiving of output and offline coupling through separate scripts. ESM-Tools facilitates Earth system modelling by providing a standardized framework to download, configure, compile, run and analyse/monitor a variety of ESMs on various HPC systems.

A variety of software and tools have been developed to assist Earth system modellers with running coupled models. However, most of them have been developed for specific purposes and lack adaptability.

- Probably closest to ESM-Tools, both by functionality and code design, is the ScriptEngine (https://pypi.org/project/scriptengine, last access: 29 June 2021) simulation task generating software developed at the Swedish Meteorological and Hydrological Institute (SMHI) primarily for the EC-Earth community. Indeed, ScriptEngine and ESM-Tools both use a combination of YAML Ain't Markup Language (YAML) (https://yaml.org/, last access: 29 June 2021) configuration files and Python methods, but while the ESM-Tools uses YAML to store the information about models, setups, etc., ScriptEngine goes more into the direction of defining a domain-specific language (DSL; see also Lawrence et al., 2018), allowing the user to describe commands and simple routines in YAML format. This approach is also very elegant and leads to a simple and user-friendly way to formulate what used to be run-scripts. To our knowledge, it is not in an advanced state yet and only contains functions for the EC-Earth model at this point. Also to our understanding, it does not contain download and compile functionality, as well as a natural concept of modularity of model components. We hope to be able to cooperate with the ScriptEngine developers in the future, as both software packages can be seen as natural extensions of each other.

- The Modular Earth Submodel System, MESSy (https://www.messy-interface.org/, last access: 20 June 2021), has been developed as an infrastructure with generalized interfaces for implementation of ESMs (Jöckel et al., 2005). The focus of MESSy, even though it comes with its own scripts and tools for compiling and running, is internal infrastructure, meaning the definition of reusable codes ("building blocks", "dwarfs", etc.) as operators, automatic management of memory layout and internal 3-D coupling of fields through all compartments of a coupled setup. MESSy and ESM-Tools do not interfere at all, as one is for the internal and the other for external infrastructure, and can in fact be used together. We already profitted from a frequent exchange and cooperation with the MESSy developers in the context of the Helmholtz project ESM and have implemented first steps towards integrating full MESSy support into ESM-Tools.

- The Modular System for Shelves and Coasts, MOSSCO (http://www.mossco.de, last access: 29 June 2021) provides a modular system for domain and process coupling of applications in the coastal ocean (Lemmen et al., 2017). It is designed to enable integrated regional coastal modelling and is targeted towards the coupling of model components with different orders of magnitude of spatial and temporal resolution. The framework allows for the seamless replacement of individual model components. In contrast to ESM-Tools, MOSSCO is a coupling framework rather than a framework for building and running ESMs.

- The Make Experiments tools, Mkexp (https://code.mpimet.mpg.de/projects, last access: 29 June 2021), contain a set of tools for preparing experiments with the Earth system models developed at the Max Planck Institute for Meteorology. It was primarily dedicated to generating run- and post-processing scripts for the Max Planck Institute Earth system model (MPI-ESM) coupled Earth system model from configuration files. One main difference between Mkexp and ESM-Tools is that the direct output of Mkexp is lengthy shell scripts, which are then submitted by the user to HPC systems. Even though Mkexp was successfully adapted to other

coupled ESMs, in our opinion it lacks the needed modularity required for optimal coupling of ESMs. The main problem of Mkexp seems to be that modellers do not use it as intended though, distributing and editing the very long and unintuitive shell scripts, rather than generating new ones. Out of this experience, we decided that ESM-Tools should not generate (complex) scripts, but rather interpret short and simple ones.

– The Earth System Modeling Framework, ESMF (http://earthsystemmodeling.org/, last access: 29 June 2021) is a standard software platform for Earth system models (Hill et al., 2004). It provides different structures for interconnecting between model components and provides a standard support library for the construction of components. The emphasis of ESMF is to ensure a standard infrastructure of component coupling and may require code adaptation in order to fit into its framework.

ESM-Tools should not be confused with the Earth System Model Evaluation Tool (ESMValTool; https://www.esmvaltool.org/, last access: 29 June 2021), which is a community diagnostics and performance metrics tool used to compare one or several models against observations or their previous versions (Righi et al., 2020; Eyring et al., 2015). ESM-Tools and ESMValTool are not related.

In contrast to other software described above, e.g. MESSy or ESMF, ESM-Tools is designed to help scientists build and run different stand-alone models as well as coupled setups without any need of code adaptation. To ensure this, all ESM components need to have a dedicated coupling already implemented. To our knowledge, there currently is no modular infrastructure that assists modellers in operating ESMs, incorporates a variety of stand-alone and coupled systems, is extendable and flexible, and is open to a larger community of researchers. ESM-Tools fills this gap by fulfilling the criteria such as user friendliness, modularity, portability, maintainability and extendability (see also Sect. 2.2.2).

This paper aims to present ESM-Tools as released in version 5.0 (December 2020); it explains the objectives of ESM-Tools, describes the development steps of the software and provides a short overview of the purpose and usage of each of the individual tools. The paper is structured as follows: In Sect. 2, ESM-Tools is described including the programming language, their structure and supported numerical models. Section 3 explains the development of the tools with focus on software and code management. The tool's application is discussed in Sect. 4. The summary including the benefits of using the tools and the experiences gained during the development is detailed in Sect. 5. Supporting information and documentation is given in Sect. A.

```
git clone https://github.com/esm-tools/esm_tools.git
cd esm_tools
./install.sh
```

Figure 1. Installation process of ESM-Tools. Some basic requirements need to be met before trying to install (like setting the locale and providing a recent version of git).

2 ESM-Tools description

The technical aspects of applying an ESM can be challenging and time consuming. This is especially true for less experienced modellers but also holds for highly experienced scientists. ESM-Tools is a software developed to reduce avoidable complexity by providing an infrastructure for obtaining and operating both stand-alone models and coupled systems.

2.1 Overview

The ESM-Tools software is divided into three major parts. The first one, called *esm_tools*, is the starting point when installing the tools and consists only of a collection of YAML configuration files containing all information on models, coupled setups, HPC systems, etc., as well as the (extensive) documentation. After cloning *esm_tools*, an installer for the other packages can be used to *pip* install the Python packages containing the actual code. The whole installation process of ESM-Tools thus consists of what can be seen in Fig. 1.

The second major part, and first important executable installed, is the *esm_master* tool that downloads, configures and compiles model components, coupled setups, libraries, etc. The tool itself is written without any reference to specific models or HPC environments but takes the required information from the *esm_tools* configuration files, i.e. in the case of the stand-alone Finite-Element/volumE Sea ice-Ocean Model (FESOM-2) (Danilov et al., 2017):

– hardware and software stack settings from a machine-dependent YAML file; e.g. in the case of *mistral.dkrz.de*, this file is located at *esm_tools/configs/machines/mistral.yaml*;

– information on how git works from *esm_tools/configs/other_software/git.yaml*;

– repository name, default branch, compile settings and necessary model-dependant environment changes for FESOM-2 from a component YAML file, in this case *esm_tools/configs/components/fesom/fesom-2.0.yaml*.

The information is collected and evaluated, and then used to either download, configure or compile the model. To install FESOM-2, it is thus sufficient to type

esm_master install-fesom-2.0,

provided one has the access privilege to do so (i.e. access to the repository used for cloning). In the case of a coupled

```yaml
general:
        setup_name: fesom
        compute_time: "00:08:00"
        initial_date: '2001-01-01'
        final_date: '2001-03-31'
        base_dir: "/path/to/experiment/data/"
        nyear: 0
        nmonth: 1
        use_venv: False

fesom:
        version: 2.0
        model_dir: "/path/to/modelcodes/fesom-2.0/"
        pool_dir: "/path/to/input/data/"
        mesh_dir: "/path/to/FESOM/meshes/mesh_CORE2_final/"
        res: CORE2
        lresume: 0
        restart_rate: 1
        restart_first: 1
        restart_unit: 'm'
        post_processing: 0
```

Figure 2. Run configuration for a short run of FESOM-2.

setup, *esm_master* first reads the YAML configuration for the coupled setup, which specifies the components used, and then the configurations for the (stand-alone) model components. In the case of a conflict between stand-alone and coupled setup settings, the information from the coupled setup configuration overwrites the stand-alone configuration.

The third and most complex major part is the *esm_runscripts* package that takes care of the whole workflow for model simulations. To use *esm_runscripts*, the user needs to provide a very short run configuration – the easiest way to provide this would be another YAML file, but we also still support shell scripts at this point, as most users are more acquainted to these. For *FESOM-2*, for example, such a run configuration might look like Fig. 2. Another runscript, in shell format, can be found in the Appendix. To each supported model component and coupled setup, a number of sample runscripts are distributed with *esm_tools*.

Similar to *esm_master*, *esm_runscripts* takes the necessary information to perform the experiment from the same YAML configuration files in *esm_tools*. To start a run, the user types

esm_runscripts myrunscript.yaml -e EXPID,

where *EXPID* is the unique name of the experiment. *esm_runscripts* will then set up the experiment folder, copy/link the needed files, apply changes to namelists, etc. and finally execute the simulation. There is a lot more to *esm_runscripts*, and some features will be described in this paper, but to cover all of them is beyond the scope of this introductory paper. The interested reader is kindly invited to have a look at the documentation on https://esm-tools.readthedocs.io/en/latest/ (last access: 29 June 2021) for more details.

Since its first release, the application of ESM-Tools has proven beneficial for the scientific modelling community. New team members have successfully used ESM-Tools to understand and run a standard ESM experiment in 1–2 d rather than 3–4 weeks prior to the availability of ESM-Tools. More experienced users have run new experiments within hours. Another benefit has been the reduction of time spent by modellers to switch to a different model or coupled system. Since ESM-Tools provides a common infrastructure for a number of different models, the workflow does not change when changing to another model or HPC system. This has proven to save modellers a considerable amount of time in the order of several days due to reduction of technical work. In this sense, ESM-Tools can make it easier for modellers to switch to other modelling systems without having to deal with periods of reduced scientific productivity.

In addition to increasing the time efficiency, ESM-Tools has assisted modellers in handling and managing model data. *esm_runscripts* manages the input and output transfer of data and log files to associated folders. This is beneficial for individual modellers as well as modelling teams. Through ESM-Tools, the modellers are able to easily exchange the path to model input data on a specific HPC. Furthermore, applying *esm_runscripts* results in a common structure for the input/output and log files/folders across model simulations. Model developers and system administrators, who have added or changed certain aspects in a model component (e.g. fixed a bug, changed configurations/parameterizations or switched pre-installed libraries with newer versions) have also used ESM-Tools to share their changes with all users in an efficient and standardized way.

2.2 Implementation and configuration

One objective of the development of ESM-Tools is to provide users with easy-to-understand functions and configuration files. To achieve this, the software architecture of ESM-Tools is structured so that all information that is mandatory for an experiment (e.g. HPC systems, input datasets) is contained in separate YAML configuration files, while the actual programme, i.e. the commands to be performed using this information, is in itself entirely independent of the models or HPC systems used. ESM-Tools functionality has been fully coded in Python. This separation of information from implementation facilitates the development and maintenance of the functions while making the tools more user friendly. Furthermore, adaption of the tools to new model configuration files and extension to include new ESMs is made easier.

Splitting information contained in the YAML configuration files from the Python implementation of the functionality in the described way has proven to be a very robust strategy to provide a modular and extendable software tool, with limited need for maintenance as users primarily edit the configuration files or user-friendly information part (written in YAML), instead of writing additional (Python) code.

Table 1. ESM-Tools Python packages.

Python package	Description
esm_master	Python functions of the executable *esm_master*, the tool to download, config and compile
esm_runscripts	Functions for the executable *esm_runscripts*, interprets runscripts, prepares and performs simulation, sorts the output, etc.
esm_version_checker	Provides the executable *esm_versions*, which helps manage/update the versions of these Python packages
esm_environment	Assembles the runtime or compile time environment needed
esm_parser	Parser for the YAML configuration, also performs list expansion, basic math, conditional parsing, etc.
esm_calendar	Basic calendar functions, much like datetime, but also works for paleo timescales
esm_plugin_manager	Package that facilitates writing of functionality as a plugin. Users and core developers can write own Python code and use a YAML configuration file to alter the flow of execution of *esm_runscripts*, while sticking to the very simple interface that a user-defined function should take a Python dictionary as input and also return this dictionary (basically the internal representation of all YAML files; this dict contains all information available at the moment of execution of the function). Figure 3 shows the order of execution of a compute job, for example, as defined in *esm_runscripts.yaml*, where additional calls could be inserted. Changes to ESM-Tools Python code are not necessary, which not only makes the development of new functionality much easier but also helps keeping the core code more stable.

```
compute:
        recipe:
                - "venv_bootstrap"
                - "_create_setup_folders"
                - "_create_component_folders"
                - "initialize_experiment_logfile"
                - "copy_tools_to_thisrun"
                - "compile_model"
                - "_copy_preliminary_files_from_experiment_to_thisrun"
                - "_show_simulation_info"
                - "create_new_files"
                - "prepare_coupler_files"
                - "add_batch_hostfile"
                - "assemble"
                - "log_used_files"
                - "_write_finalized_config"
                - "copy_files_to_thisrun"
                - "modify_namelists"
                - "modify_files"
                - "copy_files_to_work"
                - "write_simple_runscript"
                - "report_missing_files"
                - "database_entry"
                - "submit"
```

Figure 3. Section of *esm_runscripts.yaml* defining the order of execution of Python functions in a compute job. The user can easily add new (own) functions.

2.2.1 Python implementation

As a high-level, object-oriented language, Python enables algorithm flexibility and ease of programming (Pelupessy et al., 2017). Furthermore, Python is widely used by the scientific community and has many available libraries. Therefore, it is easier for the ESM-Tools users to understand the functions and contribute to the development.

We separate our (Python) code into several independent Python packages; each can be installed either using the installer distributed with the YAML configurations or the well-known Python installer *pip*.

As of version 5.0 of ESM-Tools, the main Python code consists of the packages listed in Table 1.

2.2.2 YAML configurations

The configuration files, written in YAML syntax, are used for storing all known information, including on model components and coupled setups, HPC systems and batch systems, usage of external software, configuration of ESM-Tools itself, etc. Or, to put it another way, the Python code is entirely oblivious to models, couplings, coupled setups, experiments and so on. Figure 4 shows a section of the YAML configu-

```
metadata:
        Institute: Alfred Wegener Institute
        Description:
                Multiresolution sea ice-ocean model that solves the equations
                of motion on unestructured meshes
        Publications:
                - "The Finite-volumE Sea ice-Ocean Model (FESOM2) <https://doi.org/10.5194/gmd-10-765-2017>"
                - "Scalability and some optimization of the Finite-volumE Sea ice-Ocean Model, Version 2.0 (FESOM2) <https://doi.org/10.5194/gmd-12-399
1-2019>"
```

Figure 4. Section of the *fesom-1.4* YAML configuration, containing the model metadata used to automatically assemble parts of the ESM-Tools documentation.

ration file for *fesom-1.4*, containing metadata on the model itself, which is parsed automatically by a sphinx project to assemble the documentation. This should just serve as an example to show how effective this approach is. Not only can each piece of information be used to control the compilation and execution of the models, but it can also be used to generate a complete description of an experiment (and in this sense enable provenance tracking and reproducibility) or a how-to "cookbook" for certain models. Even the flow of execution of *esm_runscripts* itself is controlled by a YAML file instead of normal Python code.

The YAML format is powerful yet easy to edit and understand by less experienced users. An extra parser was developed to correctly parse the model configuration files. This parser includes features such as (i) using only part of the parsed file (required blocks), (ii) inclusion of variable expansion, (iii) list expansion and (iv) mathematical expressions. The ability to read parts of a file (block) is useful when the modeller needs to switch off a component of the coupled system (e.g. biogeochemistry or ocean). Therefore, the same file can be used to implement an ESM with different combinations of model components. List expansion is not required but helpful in shortening the YAML configuration files. For example, if the coupled fields are not known by default but are rather defined by the user, list expansion can be used to define the names of output files of the model. Mathematical expression such as calculating the previous or next year of a run is required when preparing the input files for a consecutive run or locating restart files.

Overall, the ESM-Tools functions and configuration files fulfil the following criteria:

User-friendly: The functions and configuration files are easy to understand by the users and equally easy to edit and extend. This is mainly because the vast majority of the users do not need to be able to write Python code at all, as it is sufficient to work on the YAML configurations. Only for implementing new functionality of ESM-Tools does a modeller need to write Python functions, but even then, our self-written plugin manager can be used to work on new functionality without touching the ESM-Tools core.

Modular: The ESM-Tools software is modular itself, in more than one way. First, we treat model components independently and in the way we would treat stand-alone models. For a coupled setup such as the Alfred Wegener Institute Climate Model (AWI-CM), that means we download each of the three components (atmosphere ECHAM, ocean FESOM and coupler, OASIS) – if possible – from its own repository, then configure and compile it on its own. It is easy to change the URL of the repository or the branch to be compiled in order to not stand in the way of users who make changes in the code or who work on private development branches. In the case of coupled setups, we activate the coupling within the models by defining preprocessing options. This of course means that the coupling itself, both technically by implementing calls to a dedicated coupling software, as well as physically by adapting the parametrization of the components, needs to be established beforehand. This is a time-consuming task, and ESM-Tools should not be mistaken for a tool to couple two components – it is a tool to provide configurations for model components as well as existing coupled setups to scientists and to make the choice of components in an ESM as easy as if it were plug and play.

Using the same model repository for a component, for coupled setups and stand-alone applications, is a huge improvement for model developers who can more easily share their work with each other; it requires agreements and standards though that model developers follow in order to keep a code repository stable. An example of what *esm_runscripts* is doing exactly in the case of installing the coupled ESM *awicm-2.0* (Sidorenko et al., 2015) is shown in Fig. 5. Even though this is the strategy preferred by us and most active ESM-Tools users at the time being, it would also work to include whole coupled setups without referring to their components in a modular way.

In the same way, we write the configuration files for the runtime infrastructure as if each component would be used as a stand-alone model and just add a short configuration for each coupled setup, containing only the things that need to be changed for a component to be coupled. In this way, it becomes very easy to add new components and couplings.

The second level of modularity we take special care of is the separation of information and functionality, meaning that we have the YAML configuration files contain-

ing the information we have (on models, setups, HPC systems, version control system, etc.) and Python functions for the functionality of ESM-Tools. This implies that users/model developers who want to add, e.g. new model components, scenarios and meshes, do not need to write code at all but merely edit readable YAML lists.

Portable: The tools can run on different computer architectures.

Maintainable: The functions are short and simple enough for efficient maintenance by the developers.

Extendable: Since the system is modular, new ESMs or functions can be implemented in the tools fairly easily. For a new model component, coupled ESM, HPC system, etc., a user needs to write an additional YAML configuration. New functionality can be added by writing Python functions and using the plugin manager to schedule the execution of the function. In this way, even contributing developers who want to work on ESM-Tools functionality do not have to read and edit the ESM-Tools core, which makes it easier for the developer but also for the authors as core developers, as we can maintain the core functions independent of each other.

Performance: Using ESM-Tools is not detrimental to model performance, including high-resolution model configurations.

2.3 Structure and architecture

2.3.1 Compile time

The executable *esm_master* is a tool to download, configure and compile a variety of stand-alone models and coupled systems on different HPC systems. The command line utility provides a user-friendly interface to the underlying build environment that is provided by each model component or coupled setup. This enables the modeller to use the same syntax to install different models despite their difference in the underlying build environment, which remains unchanged as *esm_master* rather maps the commands required for each build to a common syntax. Figure 6 shows the schematic layout of the *esm_master* functioning. In this figure, models 1–3 have different building strategies (e.g. cmake, custom-made Makefile, autotools). However, the user is able to configure/compile the model without the knowledge of all the different building strategies or environments used by the different models. This results in effectively reducing the time required by the modeller to obtain a first model executable.

Again, we want to point out that the main aim of ESM-Tools is to make available easily and in a unified way what already exists, and that includes optimal HPC settings for compilation and execution of ESMs. In this sense, we collect the solutions that were already found by model developers and make sure that they become the standard setting for the given model. In principle, that configuration could be very different for each model – it turns out though that with few exceptions, most of the coupled setups and stand-alone models supported by ESM-Tools run well with the same settings. And in this way, we again can help the model developers to get an idea how to utilize an unknown model or HPC system by providing an environment setting that is known to work well for other users.

2.3.2 Runtime

The *esm_runscripts* tool is the most extensive component of ESM-Tools in terms of functionality. This tool consists of optimized implementations of all the functionalities to set up all required phases to run an ESM simulation. The user only needs to provide a short script that includes experiment-specific definitions, while the *esm_runscripts* execute all the phases of a simulation in the correct order (see Fig. 7).

The first task of this tool is to prepare the simulation setup. This includes creating the folder structure of the run, allocating the run files/folders, preparing/allocating initial and forcing files and preparing the model input files via the esm_parser package (see Sect. 2.2). The next task is to run the model by submitting the executable to an HPC batch system or job scheduler. During the execution of the run, the tool monitors the simulation, creates log files and provides feedback on the progress of the run. After the completion of the run, the log files and output data are moved to designated folders. Finally, the post-processing of the output data, including preparation of the consequent run, is carried out.

2.3.3 Online monitoring

The *esm_viz* is a command line tool to schedule automatic monitoring of Earth system model simulations. A key benefit here is the ability to visualize particular aspects of a simulation as it progresses. As with the remainder of ESM-Tools, *esm_viz* is configured with a YAML file. Plots representing climatological averages and time series can be automatically generated, which are interactively displayed in the browser and can be downloaded in a high-quality format for inclusion in other scientific outputs, such as conference posters, papers or talks. Additionally, *esm_viz* gives an overview of a job, with an estimated completion time, approximations of queuing time and real model throughput and links to relevant log files. Notably, *esm_viz* runs on a separate computer to the supercomputer and is scheduled via a cronjob, allowing for independent monitoring and reporting even if the main computer running the simulation goes offline. In this case,

```
[a270058@mlogin108% esm_master install-awicm-2.0
    Executing commands in this order:
         mkdir -p awicm-2.0
         cd awicm-2.0
         git clone -b 2.8 https://a270058@gitlab.dkrz.de/modular_esm/oasis3-mct.git oasis
         git clone -b 6.3.04p1 https://a270058@gitlab.dkrz.de/modular_esm/echam6.git echam-6.3.04p1
         git clone -b 2.0.2 https://a270058@gitlab.dkrz.de/FESOM/fesom2.git fesom-2.0
         sed -i '/set(FESOM_COUPLED/s/OFF/ON/g' fesom-2.0/CMakeLists.txt
         sed -i '/ECHAM6_COUPLED/s/OFF/ON/g' echam-6.3.04p1/CMakeLists.txt
         cd oasis
         mkdir -p build; cd build; cmake ..;   make -j 1; mkdir -p ../include/; cp lib/psmile/libpsmile.a lib/psmile/mct/libmct.a lib/psmile/mct/mpeu/l
ibmpeu.a lib/psmile/scrip/libscrip.a ../lib; cp lib/psmile/mod_oasis*mod ../include
         cd ..
         mkdir -p ./lib
         cp  oasis/build/lib/psmile/libpsmile.a  lib
         cp  oasis/build/lib/psmile/mct/libmct.a  lib
         cp  oasis/build/lib/psmile/mct/mpeu/libmpeu.a  lib
         cp  oasis/build/lib/psmile/scrip/libscrip.a  lib
         cd echam-6.3.04p1
         mkdir -p build; cd build; cmake ..;   make install -j `nproc --all`
         cd ..
         mkdir -p ./bin
         cp  echam-6.3.04p1/src/echam/bin/echam6  bin
         cd fesom-2.0
         mkdir -p build; cd build; cmake ..;   make install -j `nproc --all`
         cd ..
         mkdir -p ./bin
         cp  fesom-2.0/bin/fesom.x  bin
         cd ..
```

Figure 5. Output of *esm_master install-awicm-2.0*, listing the steps performed by *esm_master* to download, configure and compile the three components *echam-6.3.04p1*, *fesom-2* and *oasis3mct*. Note that preprocessor flags are used to change the model components from the stand-alone to the coupled configurations.

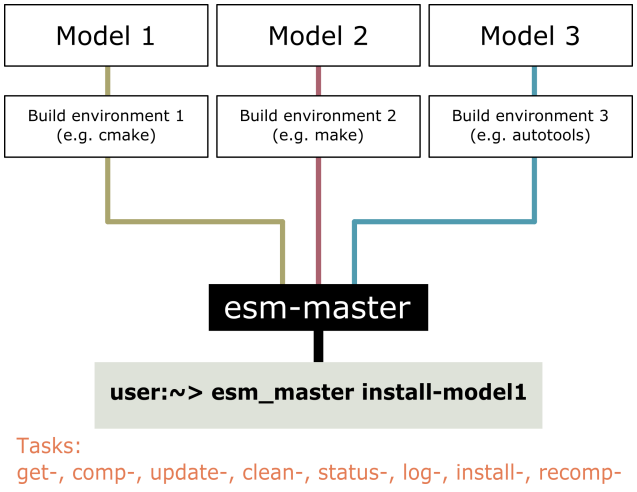

Figure 6. Schematic of the *esm_master* tool, which provides generic building commands that map to each model-component-specific building environment. Further generic tasks are comp-, update-, clean-, status-, log-, install- and recomp-.

esm_viz also provides information about the last available state of the simulation. For all further information, we refer the reader to a future publication.

2.3.4 Graphical user interface

A prototype graphical user interface (GUI) was developed for a previous version (v2.0) of ESM-Tools (see also Fig. 8) and is, to our knowledge, the first developed GUI that incorporates multiple ESMs. The development of the GUI was possible due to the modular layout of the repositories of the models used by ESM-Tools, as well as treating each component separately in what has become the YAML configurations. Users are able to download and compile the chosen setup or models, which are selected from the standard or customer config tab.

2.4 Supported models, coupled systems and HPC environments

The current version of ESM-Tools contains configuration files for four ocean models, three atmosphere models, two biogeochemistry models, an ice sheet model, an isostatic adjustment model, a hydrology model and a land-surface model (see Tables 2 and 3 for details). It must be clear that using ESM-Tools, which is under GPLv2 open-source license, does not include licenses to these models, and the download tool (*esm_master*) will only work if the corresponding paths are set to valid repositories that the user actually has access to.

Two different batch systems are currently supported by the tools: Slurm (https://slurm.schedmd.com/documentation.html, last access: 29 June 2021) and Moab (http://docs.adaptivecomputing.com/mwm/7-1-3/help.htm, last access: 29 June 2021).

ESM-Tools, along with the available model configurations, can be readily run on several German HPC centres, such as the Norddeutsche Verbund für Hoch- und Höchstleistungsrechnen (HLRN), the German Climate Computing Center (DKRZ), the Jülich Supercomputing Centre (JSC) and the Alfred Wegener Institute supercomputer. The run systems and modules on these machines are regularly updated or even completely changed. Therefore, the relevant ESM-Tools machine files are also adapted accordingly by the ESM-Tools support team. This leads to the modellers saving a consider-

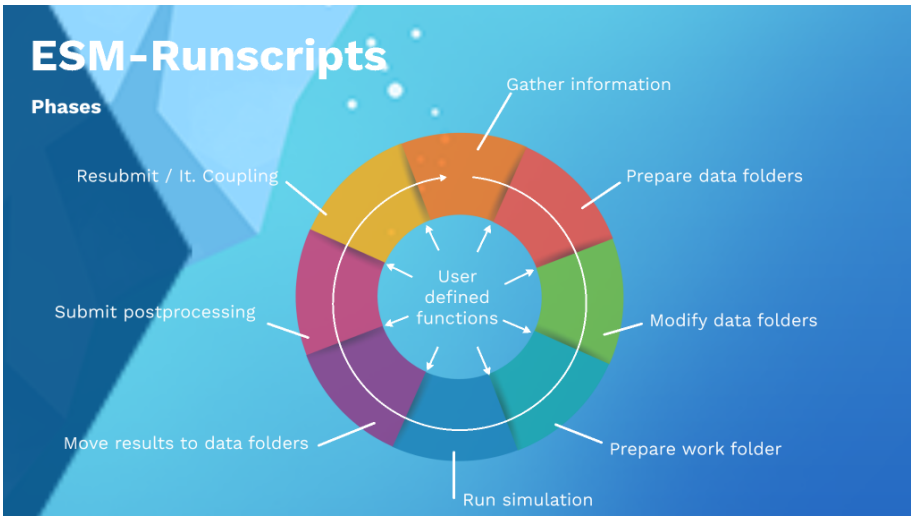

Figure 7. Schematic of the tasks carried out by *esm_runscripts*. In addition to standard job phases, users and developers have the possibility to add their own Python code via a plugin manager, which can be executed between the pre-defined function calls without the need to edit ESM-Tools core functionality (denoted as "user defined functions").

Table 2. Supported model components.

Model component	Model type	Reference
FESOM 1.4	Ocean	Danilov et al. (2004)
FESOM 2.0	Ocean	Danilov et al. (2017)
MPIOM	Ocean	Jungclaus et al. (2013)
NEMO	Ocean	Madec and the NEMO team (2008)
ECHAM	Atmosphere	Stevens et al. (2013)
ICON-A	Atmosphere	Giorgetta et al. (2018)
OpenIFS	Atmosphere	https://www.ecmwf.int/en/research/projects/openifs (last access: 29 June 2021)
ReCom	Biogeochemistry	Hauck et al. (2013); Schourup-Kristensen et al. (2018)
PISM	Ice sheet	Winkelmann et al. (2011)
Vilma	Isostatic adjustment	Martinec (2000)
HD-model	Hydrology	Hagemann and Dümenil (1998)
JSBach	Land surface	Raddatz et al. (2007)

able amount of time needed for understanding the new system and adapting their files.

3 ESM-Tools development

3.1 Software and code management

The first two versions of ESM-Tools were released in September 2018 and March 2019, respectively. In these versions, the functions were mainly coded in Bash, and the code was mixed with model configuration. With the growth of the tools and the addition of various ESMs, maintaining the functions written in Bash became non-trivial and time consuming. This was mainly due to the Bash language not being object oriented in nature. Therefore, in the third version of the tools, a complete migration of the functions to Python was carried out. The ESM-Tools software is currently hosted and developed on GitHub (see also Code availability). The ESM-Tools core development team consists of four scientists and scientific programmers from two groups (Climate Dynamics and Paleoclimate Dynamics) at the AWI. These developers take care of the strategic planning and implementation of new main features, as well as organizing regular developer meetings, workshops for new users, etc. On a second tier, more than 30 scientists from AWI, but also from GEOMAR Helmholtz-Zentrum für Ozeanforschung Kiel and Helmholtz-Zentrum Potsdam Deutsches GeoForschungsZentrum (GFZ), authored contributions to the tool, which makes it an active open-source community.

Special focus has been placed upon the documentation, for which we try to provide everything that is possible – from a standard user manual, command line help, man pages, etc.

Table 3. Supported coupled setups.

Coupled setup	Model components	Coupler	Reference
AWI-CM-v1.0	FESOM-1.4, ECHAM-6.3	OASIS3-MCT	(Sidorenko et al., 2015; Rackow et al., 2018)
AWI-CM-v2.0	FESOM-2.0, ECHAM-6.3	OASIS3-MCT	(Sidorenko et al., 2019)
AWI-CM-RECOM	AWI-CM-v2.0, REcoM2	OASIS3-MCT	(in preparation, 2021)
AWI-CM-PISM	AWI-CM-v2.0, PISM	OASIS3-MCT, SCOPE?	(Gierz et al., in preparation, 2021)
OpenIFS-FESOM	FESOM-2.0, OpenIFS 43	OASIS3-MCT	(Streffing et al., in preparation, 2021)
OpenIFS-NEMO	NEMO-3.6, OpenIFS 43	OASIS3-MCT	(in preparation, 2021)
FOCI	NEMO-3.6, ECHAM-6.3	OASIS3-MCT	(Matthes et al., in preparation, 2021)
MPI-ESM	MPIOM, ECHAM-6.3	OASIS3-MCT	(Giorgetta et al., 2013)
MPI-ESM-PISM	MPI-ESM, PISM	OASIS3-MCT	(Ziemen et al., 2019)

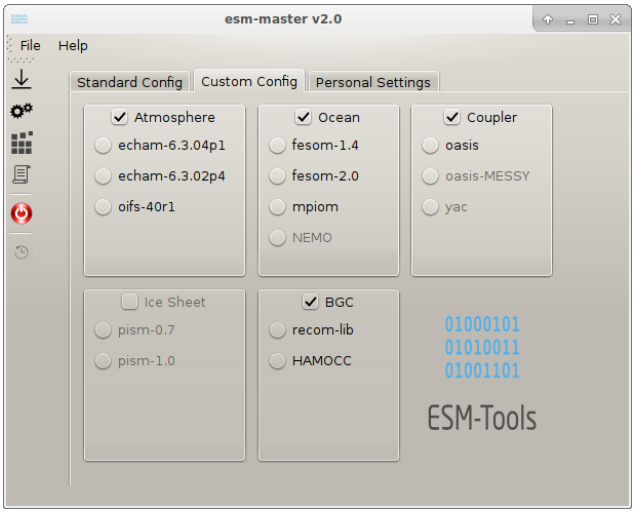

Figure 8. GUI for ESM-Tools, which helps to download, compile and configure selected models.

up to a website (https://www.esm-tools.net, last access: 29 June 2021) containing course material and a YouTube channel (ESM-Tools) with tutorial videos.

To efficiently organize the development of the model, agile software development is applied (Dingsøyr et al., 2012). Since software requirements in scientific projects are often unclear at the start (Wieters and Fritzsch, 2018), agile software development allows for the adaption of the software to the user requirements during the development. Regular weekly and monthly meetings are organized to manage the development. During the weekly meetings, the lead development team discusses the progress of the tools and plans the timeline for the next week. During monthly meetings, the larger development team, including members of other groups from partner institutes, discusses the status of the development, issues and problems, and newly implemented features. Furthermore, requirements for future developments, collected from users, are also reviewed. In this way, the road map for future releases is continuously adapted to meet the target of two software releases per year.

3.2 Contributing to the development

The development of ESM-Tools consists, to a large extent, of the contribution of users who develop their own extensions to the existing code. To extend the list of supported models, coupled systems and HPC systems (see also Sect. 2.4), users are encouraged to develop new features and functionalities according to their requirements. This can be done by contributing to the development on GitHub (e.g. by pull requests). Furthermore, users can suggest a "wishlist" of missing functionalities (e.g. via issue tracker). The ESM-Tools developers then work in close collaboration with the users to implement and test the new features. Regular workshops are also organized for users and developers. In addition to the development, the most important contribution by users is reporting model errors and bugs, which are then resolved by the core developers. Apart from adding to the Python code for more ESM-Tools core functionality, the main contribution users have provided in the past was the addition of additional model components, coupled setups or experiment parameters into the YAML configuration. As an example, to include an additional model component, the following steps should be followed:

- Make the code available on a git or svn server. That should be a standard anyway.
- Create a pool directory on the HPC server, containing the needed forcing, input and restart data.
- Add standard namelists/configuration files to *esm_tools*. These can be templates changed by the tools later on.
- Create a model YAML configuration file containing the information about how to download and compile the model. As an example, in Fig. 9, we show the entries controlling download and compilation for *fesom-1.4*. This should enable all the functionality of *esm_master*.
- Add the information about needed files, e.g. forcing data, naming of restart files, namelists, etc.

```
# FESOM YAML CONFIGURATION FILE
#

model: fesom
branch: master
version: "1.4"
type: ocean

available_versions:
- 1.4-esm_interface
- '1.4'
- 1.4-recom-awicm
choose_version:
  '1.4-recom-awicm':
    destination: fesom-1.4
    branch: co2_coupling
  '1.4-esm_interface':
    destination: fesom-1.4
    branch: using_esm-interface

git-repository: https://gitlab.dkrz.de/modular_esm/fesom-1.4.git
install_bins: bin/fesom
clean_command: ${defaults.clean_command}
comp_command: mkdir -p build; cd build; cmake ..;   make install
```

Figure 9. Excerpt from *fesom.yaml* with the information on how to compile and download *fesom-1.4*.

```
namelist_changes:
        namelist.config:
                clockinit:
                        yearnew: "${initial_date!syear}"
                calendar:
                        include_fleapyear: "${leapyear}"
                paths:
                        ForcingDataPath: "${forcing_data_dir}"
                        MeshPath: "${mesh_dir}"
                        OpbndPath: "${opbnd_dir}"
                        ClimateDataPath: "${climate_data_dir}"
                        TideForcingPath: "${tide_forcing_dir}"
                        ResultPath: "${work_dir}"
                timestep:
                        step_per_day: "${steps_per_day}"
                        run_length: "${restart_rate}"
                        run_length_unit: "${restart_unit}"
                inout:
                        restartflag: "${restart_flag}"
                        output_length: "${restart_rate}"
                        output_length_unit: "${restart_unit}"
                        restart_length: "remove_from_namelist"
                        restart_length_unit: "remove_from_namelist"
                mesh_def:
                        part_format: "remove_from_namelist"
```

Figure 10. Excerpt from *fesom.yaml*, specifying how to change the namelist templates during the preparation phase of a run.

- Add at least the parameters for one standard experiment for testing, including namelist changes, etc. In Fig. 10, see an example again from *fesom-1.4*.
- Add a sample runscript for this experiment, preferably in YAML format.

As this is just a short overview and a detailed how-to section is beyond the scope of this paper, we kindly invite the interested reader to have a look on our quite extensive documentation, both on GitHub as well as our web page, or use the existing configuration files as a source of inspiration.

4 Applications

Currently, around 75 scientists at AWI and partner institutes are registered users of ESM-Tools. This includes mostly model users but also model developers/maintainers and sys-

tem administrators. The tools are actively used to run the AWI Climate Model (AWI-CM). In fact, all contributions of AWI to the Coupled Model Intercomparison Project phase 6 (CMIP6) were produced using ESM-Tools (Semmler et al., 2020). The application of the tools has resulted in a more efficient exchange of model configurations and input and output model data between modellers and scientists.

5 Discussion

ESM-Tools provides a standardized way of working with ESMs. In addition to the benefits of applying ESM-Tools mentioned above, the development of ESM-Tools has led to various new insights. From a technical perspective, using an object-oriented language for such a software tool has proven to be essential for efficient expansion and maintenance of the tools themselves. The previous versions of ESM-Tools, written in Bash, suffered from lengthy functions as well as entangled functions and configuration files that were hard to maintain. The separation of configuration files and ESM-Tools functions in v3.0 as well as using a high-level language has made ESM-Tools more modular and easy to maintain.

From a community-building perspective, the development has brought together scientists and modellers from various backgrounds and institutes, resulting in a new community of modellers and scientific programmers in the geosciences. Through developing and using ESM-Tools, programmers and modellers have efficiently shared their knowledge on best coding practices and learned to collaborate with each other more closely. In conclusion, ESM-Tools has developed into an active open-source community, making it much easier for modellers, model developers and system administrators to obtain/provide access to, run and maintain Earth system models in a standardized way.

Apart from the things ESM-Tools can do to simplify model usage for the modellers, which were discussed already extensively in this paper, it is also important to point out that ESM-Tools is *not* any of the following:

- A coupling software (like OASIS or YAC).
- A tool that automatically couples model components – even though our tools, especially the GUI, look as if it were coupling model components in a "plug-and-play" way, that is of course not the case. Coupling is a complex and time-consuming task, and behind each of the couplings supported in ESM-Tools, several personal months of technical work, parametrization and fine tuning are hidden. ESM-Tools only makes the already established couplings as easy available as if it were plug and play.
- A replacement for model build systems/model runscripts – we understand our tools as alternatives, not replacements, meaning trying ESM-Tools does not influence already existing solutions, and the user can switch back to them at any time.
- A coding standard/collection of generic interfaces/DSL. Using ESM-Tools does not involve changes to the model codes at all, as we provide an external infrastructure.

  Generic interfaces are the main purpose of a second software, called *esm-interface*, that is currently under development. It provides a library of generic procedures to enable a modular approach to ESM coupling. This software will be discussed in a separate publication (in preparation).

Appendix A: Supporting information and documentation

ESM-Tools is well documented and therefore easy to install and understand. The documents are constantly updated by the model developers. Users are also encouraged to contribute to the documentation.

In order to facilitate the use of ESM-Tools, regular workshops are conducted for users and developers. The users can also report bugs/problems via GitHub, which are then solved by developers. The ESM-Tools web page (https://www.esm-tools.net/, last access: 29 June 2021) provides essential information on ESM-Tools. As an outreach, monthly newsletters containing report on new features, bug fixes and upcoming events regarding ESM-Tools are released. The users can also refer to Twitter (@ToolsEsm #ToolsEsm) for further updates.

A1 Example ESM-Tools runscript

```
set -e


setup_name="awicm"
#check=1

account=ab0995
compute_time="00:15:00"
################################################################################

INITIAL_DATE_awicm=2000-01-01       # Initial exp. date
FINAL_DATE_awicm=2000-02-01         # Final date of the experiment

awicm_VERSION="CMIP6"
POST_PROCESSING_awicm=0
SCENARIO_awicm="PI-CTRL"

RES_fesom=CORE2

MODEL_DIR_awicm=${HOME}/esm_yaml/awicm-CMIP6/
BASE_DIR=/work/ollie/dbarbi/esm_yaml_test/

POOL_DIR_fesom=/work/ollie/pool/FESOM/
MESH_DIR_fesom=/work/ollie/pool/FESOM/meshes_default/core/

NYEAR_awicm=0           # Number of years per run
NMONTH_awicm=1          # Number of months per run

LRESUME_echam=0
LRESUME_fesom=0
LRESUME_oasis3mct=0

RESTART_RATE_fesom=1
RESTART_FIRST_fesom=1
RESTART_UNIT_fesom='m'


further_reading_fesom="fesom_output_control.yaml"


################################################################################
load_all_functions
general_do_it_all $@
```

Figure A1. Example runscript, coupled model setup AWICM-2.0, shell style.

A2 Example YAML configuration file

```yaml
##############################################################################################
######################### AWICM 1 YAML CONFIGURATION FILE ####################################
##############################################################################################

general:
        model: awicm
        model_dir: ${esm_master_dir}/awicm-${version}

        coupled_setup: True

        include_models:
                - echam
                - fesom
                - oasis3mct

        version: "1.1"
        scenario: "PI-CTRL"
        resolution: ${echam.resolution}_${fesom.resolution}
        postprocessing: false
        post_time: "00:05:00"
        choose_general.resolution:
                T63_CORE2:
                        compute_time: "02:00:00"
                T63_REF87K:
                        compute_time: "02:00:00"
                T63_REF:
                        compute_time: "02:00:00"


##############################################################################################
########### necessary changes to submodels compared to standalone setups #################
##############################################################################################

echam:
        restart_firstlast: "first"
        namelist_changes:
                namelist.echam:
                        runctl:
                                lcouple: .true.
        adj_input_dir: "${fesom.mesh_dir}/tarfiles${echam.resolution}/input/echam6"
        model_dir: ${general.model_dir}/echam-${echam.version}
        setup_dir: ${general.model_dir}
        ocean_resolution: "${fesom.resolution}"
        remove_forcing_files:
                - sst
                - sic
        version: "6.3.04p1"

        choose_general.resolution:
                T63_CORE2:
                        nproca: 24
                        nprocb: 18
                T63_REF87K:
                        nproca: 24
                        nprocb: 18
                T63_REF:
                        nproca: 24
                        nprocb: 18


##############################################################################################


jsbach:
        #dynveg_file_ending: ""
        adj_input_dir: "${fesom.mesh_dir}/tarfiles${echam.resolution}/input/jsbach"
        namelist_changes:
                namelist.jsbach:
                        hydrology_ctl:
                                gethd: "remove_from_namelist"
                                puthd: "remove_from_namelist"
        version: "3.20"
```

Figure A2. Example of YAML configuration file, coupled model setup AWICM-2.0, part 1 of 2.

```yaml
fesom:
        choose_general.version:
                1.1:
                        version: "1.4"
                CMIP6:
                        version: "1.4"
                2.0:
                        version: "2.0"
        choose_general.resolution:
                T63_CORE2:
                        nproc: 288
                T63_REF87K:
                        nproc: 216
                T63_REF:
                        nproc: 128

        opbnd_dir: ""
        tide_forcing_dir: ""
        forcing_data_dir: ""
        model_dir: ${general.model_dir}/fesom-${fesom.version}
        setup_dir: ${general.model_dir}

        add_namelist_changes:
                namelist.oce:
                        boundary:
                                restore_s_surf: 0.0



##############################################################################################

oasis3mct:
        model_dir: ${general.model_dir}/oasis

        process_ordering:
                - fesom
                - echam

        a2o_lag: "${echam.time_step}"
        o2a_lag: "${fesom.time_step}"
        a2o_seq: 2

        coupling_time_step: 3600
        coupling_target_fields:
                o2a_flux:
                        - 'sst_atmo:sit_atmo:sie_atmo <--distwgt-- sst_feom:sit_feom:sie_feom'
                        - 'snt_atmo <--distwgt-- snt_feom'

                a2o_flux:
                        - 'taux_oce:tauy_oce:taux_ico:tauy_ico <--bicubic-- taux_atm:tauy_atm:taux_ica:tauy_ica'
                        - 'prec_oce <--distwgt-- prec_atm'
                        - 'snow_oce <--distwgt-- snow_atm'
                        - 'evap_oce <--distwgt-- evap_atm'
                        - 'subl_oce <--distwgt-- subl_atm'
                        - 'heat_oce <--distwgt-- heat_atm'
                        - 'heat_ico <--distwgt-- heat_ica'
                        - 'heat_swo <--distwgt-- heat_swa'
                        - 'hydr_oce <--distwgt-- hydr_atm'


        coupling_directions:
                'feom->atmo':
                        lag: ${o2a_lag}
                        seq: 2
                'atmo->feom':
                        lag: ${a2o_lag}
                        seq: ${a2o_seq}


        coupling_methods:
                distwgt:
                        name: distwgt
                        bins: 15
                        other_number: 6
                bicubic:
                        name: bicubic
                        bins: 15
```

Figure A3. Example of YAML configuration file, coupled model setup AWICM-2.0, part 2 of 2.

Code availability. ESM-Tools is open-source software developed at the Alfred Wegener Institute Helmholtz Centre for Polar and Marine Research and licensed under a modified GPLv2 licence currently hosted on https://github.com/esm-tools/esm_tools.git (esm-tools, 2021), with a mirror on https://gitlab.awi.de/esm_tools/esm_tools.git (last access: 29 June 2021). Please contact the main developers (dirk.barbi@awi.de, miguel.andres-martinez@awi.de, deniz.ural@awi.de) for development access. The code's DOI is https://doi.org/10.5281/zenodo.4899741 (Barbi et al., 2021).

Author contributions. DB is the lead developer of ESM-Tools; NW, MAM and DU are the supporting developers of ESM-Tools and contributed to the development, documentation, branches management and user support; PG is the supporting developer of ESM-Tools and contributed to the development of parts of the software, documentation, Python implementation, branch management and user support; FC contributed to software development in Python, user support and documentation management; SK developed the GUI's first version; LC contributed to team management and software project management.

Competing interests. The authors declare that they have no conflict of interest.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Acknowledgements. We especially want to thank Thomas Jung and Jan Streffing (AWI), Joakim Skjellson and Sebastian Wahl (GEOMAR) for major contributions and important feedback, Jaroslav Piwonski (University of Kiel) for helping us to organize the reimplementation in Python/YAML, and Tido Semmler, Christopher Danek and Christian Stepanek (all at AWI) for lots of testing and bug reporting.

Review statement. This paper was edited by Ignacio Pisso and reviewed by Chuncheng Guo and two anonymous referees.

Financial support. The work described in this paper has received funding from the Initiative and Networking Fund of the Helmholtz Association through the project "Advanced Earth System Modelling Capacity (ESM)".

The article processing charges for this open-access publication were covered by the Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research (AWI).

References

Barbi, D., Gierz, P., Wieters, N., Cristini, L., Streffing, J., Andrés-Martinez, M., Kjellsson, J., Wahl, S., and Ural, D.: esm-tools/esm_tools: Release 5.1 (Version v5.1.6), Zenodo [data set], https://doi.org/10.5281/zenodo.4899741, 2021.

Bauer, P., Thorpe, A., and Brunet, G.: The quiet revolution of numerical weather prediction, Nature, 525, 47–55, https://doi.org/10.1038/nature14956, 2015.

Danilov, S., Kivman, G., and Schröter, J.: A finite-element ocean model: principles and evaluation, Ocean Model., 6, 125–150, 2004.

Danilov, S., Sidorenko, D., Wang, Q., and Jung, T.: The Finite-volumE Sea ice–Ocean Model (FESOM2), Geosci. Model Dev., 10, 765–789, https://doi.org/10.5194/gmd-10-765-2017, 2017.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B.: A decade of agile methodologies: Towards explaining agile software development, J. Syst. Softw., 85, 1213–1221, https://doi.org/10.1016/j.jss.2012.02.033, 2012.

esm-tools: esm_tools, GitHub, available at: https://github.com/esm-tools/esm_tools.git, last access: 29 June 2021.

Eyring, V., Righi, M., Lauer, A., Evaldsson, M., Wenzel, S., Jones, C., Anav, A., Andrews, O., Cionni, I., Davin, E. L., Deser, C., Ehbrecht, C., Friedlingstein, P., Gleckler, P., Gottschaldt, K.-D., Hagemann, S., Juckes, M., Kindermann, S., Krasting, J., Kunert, D., Levine, R., Loew, A., Mäkelä, J., Martin, G., Mason, E., Phillips, A. S., Read, S., Rio, C., Roehrig, R., Senftleben, D., Sterl, A., van Ulft, L. H., Walton, J., Wang, S., and Williams, K. D.: ESMValTool (v1.0) – a community diagnostic and performance metrics tool for routine evaluation of Earth system models in CMIP, Geosci. Model Dev., 9, 1747–1802, https://doi.org/10.5194/gmd-9-1747-2016, 2016.

Giorgetta, M. A., Jungclaus, J., Reick, C. H., Legutke, S., Bader, J., Böttinger, M., Brovkin, V., Crueger, T., Esch, M., Fieg, K., Glushak, K., Gayler, V., Haak, H., Hollweg, H.-D., Ilyina, T., Kinne, S., Kornblueh, L., Matei, D., Mauritsen, T., Mikolajewicz, U., Mueller, W., Notz, D., Pithan, F., Raddatz, T., Rast, S., Redler, R., Roeckner, E., Schmidt, H., Schnur, R., Segschneider, J., Six, K. D., Stockhause, M., Timmreck, C., Wegner, J., Widmann, H., Wieners, K.-H., Claussen, M., Marotzke, J., and Stevens, B.: Climate and carbon cycle changes from 1850 to 2100 in MPI-ESM simulations for the Coupled Model Intercomparison Project phase 5, J. Adv. Model. Earth Sy., 5, 572–597, https://doi.org/10.1002/jame.20038, 2013.

Giorgetta, M. A., Brokopf, R., Crueger, T., Esch, M., Fiedler, S., Helmert, J., Hohenegger, C., Kornblueh, L., Köhler, M., Manzini, E., Mauritsen, T., Nam, C., Raddatz, T., Rast, S., Reinert, D., Sakradzija, M., Schmidt, H., Schneck, R., Schnur, R., Silvers, L., Wan, H., Zängl, G., and Stevens, B.: ICON-A, the Atmosphere Component of the ICON Earth System Model: I. Model Description, J. Adv. Model. Earth Sy., 10, 1613–1637, https://doi.org/10.1029/2017MS001242, 2018.

Hagemann, S. and Dümenil, L.: A parametrization of the lateral waterflow for the global scale, Clim. Dynam., 14, 17–31, 1998.

Hauck, J., Völker, C., Wang, T., Hoppema, M., Losch, M., and Wolf-Gladrow, D. A.: Seasonally different carbon flux changes in the Southern Ocean in response to the southern annular mode, Global Biogeochem. Cy., 27, 1236–1245, https://doi.org/10.1002/2013GB004600, 2013.

Hill, C., DeLuca, C., Balaji, Suarez, M., and Da Silva, A.: The architecture of the Earth System Modeling Framework, Comput. Sci. Eng., 6, 18–28, https://doi.org/10.1109/MCISE.2004.1255817, 2004.

Jöckel, P., Sander, R., Kerkweg, A., Tost, H., and Lelieveld, J.: Technical Note: The Modular Earth Submodel System (MESSy) – a new approach towards Earth System Modeling, Atmos. Chem. Phys., 5, 433–444, https://doi.org/10.5194/acp-5-433-2005, 2005.

Jungclaus, J., Fischer, N., Haak, H., Lohmann, K., Marotzke, J., Matei, D., Mikolajewicz, U., Notz, D., and Von Storch, J.: Characteristics of the ocean simulations in the Max Planck Institute Ocean Model (MPIOM) the ocean component of the MPI-Earth system model, J. Adv. Model. Earth Sy., 5, 422–446, 2013.

Lawrence, B. N., Rezny, M., Budich, R., Bauer, P., Behrens, J., Carter, M., Deconinck, W., Ford, R., Maynard, C., Mullerworth, S., Osuna, C., Porter, A., Serradell, K., Valcke, S., Wedi, N., and Wilson, S.: Crossing the chasm: how to develop weather and climate models for next generation computers?, Geosci. Model Dev., 11, 1799–1821, https://doi.org/10.5194/gmd-11-1799-2018, 2018.

Lemmen, C., Hofmeister, R., Klingbeil, K., Nasermoaddeli, M. H., Kerimoglu, O., Burchard, H., Kösters, F., and Wirtz, K. W.: Modular System for Shelves and Coasts (MOSSCO v1. 0)-a flexible and multi-component framework for coupled coastal ocean ecosystem modelling, arXiv preprint, arXiv:1706.04224, 2017.

Madec, G. and the NEMO team: NEMO ocean engine, Note du Pole 'de modelisation', Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN 1288-1619, 2008.

Martinec, Z.: Spectral–finite element approach to three-dimensional viscoelastic relaxation in a spherical earth, Geophys. J. Int., 142, 117–141, https://doi.org/10.1046/j.1365-246x.2000.00138.x, 2000.

Pelupessy, I., van Werkhoven, B., van Elteren, A., Viebahn, J., Candy, A., Portegies Zwart, S., and Dijkstra, H.: The Oceanographic Multipurpose Software Environment (OMUSE v1.0), Geosci. Model Dev., 10, 3167–3187, https://doi.org/10.5194/gmd-10-3167-2017, 2017.

Rackow, T., Goessling, H. F., Jung, T., Sidorenko, D., Semmler, T., Barbi, D., and Handorf, D.: Towards multi-resolution global climate modeling with ECHAM6-FESOM. Part II: climate variability, Clim. Dynam., 50, 2369–2394, 2018.

Raddatz, T. J., Reick, C. H., Knorr, W., Kattge, J., Roeckner, E., Schnur, R., Schnitzler, K.-G., Wetzel, P., and Jungclaus, J.: Will the tropical land biosphere dominate the climate–carbon cycle feedback during the twenty-first century?, Clim. Dynam., 29, 565–574, https://doi.org/10.1007/s00382-007-0247-8, 2007.

Righi, M., Andela, B., Eyring, V., Lauer, A., Predoi, V., Schlund, M., Vegas-Regidor, J., Bock, L., Brötz, B., de Mora, L., Diblen, F., Dreyer, L., Drost, N., Earnshaw, P., Hassler, B., Koldunov, N., Little, B., Loosveldt Tomas, S., and Zimmermann, K.: Earth System Model Evaluation Tool (ESMValTool) v2.0 – technical overview, Geosci. Model Dev., 13, 1179–1199, https://doi.org/10.5194/gmd-13-1179-2020, 2020.

Schourup-Kristensen, V., Wekerle, C., Wolf-Gladrow, D. A., and Völker, C.: Arctic Ocean biogeochemistry in the high resolution FESOM 1.4-REcoM2 model, Prog. Oceanogr., 168, 65–81, https://doi.org/10.1016/j.pocean.2018.09.006, 2018.

Schulthess, T. C., Bauer, P., Wedi, N., Fuhrer, O., Hoefler, T., and Schär, C.: Reflecting on the Goal and Baseline for Exascale Computing: A Roadmap Based on Weather and Climate Simulations, Comput. Sci. Eng., 21, 30–41, https://doi.org/10.1109/MCSE.2018.2888788, 2019.

Semmler, T., Danilov, S., Gierz, P., Goessling, H. F., Hegewald, J., Hinrichs, C., Koldunov, N., Khosravi, N., Mu, L., Rackow, T., Sein, D. V., Sidorenko, D., Wang, Q., and Jung, T.: Simulations for CMIP6 With the AWI Climate Model AWI-CM-1-1, J. Adv. Model. Earth Sy., 12, e2019MS002009, https://doi.org/10.1029/2019MS002009, 2020.

Sidorenko, D., Rackow, T., Jung, T., Semmler, T., Barbi, D., Danilov, S., Dethloff, K., Dorn, W., Fieg, K., Gößling, H. F., Handorf, D., Harig, S., Hiller, W., Juricke, S., Losch, M., Schröter, J., Sein, D. V., and Wang, Q.: Towards multi-resolution global climate modeling with ECHAM6–FESOM. Part I: model formulation and mean climate, Clim. Dynam., 44, 757–780, 2015.

Sidorenko, D., Goessling, H., Koldunov, N., Scholz, P., Danilov, S., Barbi, D., Cabos, W., Gurses, O., Harig, S., Hinrichs, C., Juricke, S., Lohmann, G., Losch, M., Mu, L., Rackow, T., Rakowsky, N., Sein, D., Semmler, T., Shi, X., Stepanek, C., Streffing, J., Wang, Q., Wekerle, C., Yang, H., and Jung, T.: Evaluation of FESOM2.0 Coupled to ECHAM6.3: Preindustrial and HighResMIP Simulations, J. Adv. Model. Earth Sy., 11, 3794–3815, 2019.

Stevens, B., Giorgetta, M., Esch, M., Mauritsen, T., Crueger, T., Rast, S., Salzmann, M., Schmidt, H., Bader, J., Block, K., Brokopf, R., Fast, I., Kinne, S., Kornblueh, L., Lohmann, U., Pincus, R., Reichler, T., and Roeckner, E.: Atmospheric component of the MPI-M earth system model: ECHAM6, J. Adv. Model. Earth Sy., 5, 146–172, 2013.

Wieters, N. and Fritzsch, B.: Opportunities and limitations of software project management in geoscience and climate modelling, Adv. Geosci., 45, 383–387, https://doi.org/10.5194/adgeo-45-383-2018, 2018.

Winkelmann, R., Martin, M. A., Haseloff, M., Albrecht, T., Bueler, E., Khroulev, C., and Levermann, A.: The Potsdam Parallel Ice Sheet Model (PISM-PIK) – Part 1: Model description, The Cryosphere, 5, 715–726, https://doi.org/10.5194/tc-5-715-2011, 2011.

Ziemen, F. A., Kapsch, M.-L., Klockmann, M., and Mikolajewicz, U.: Heinrich events show two-stage climate response in transient glacial simulations, Clim. Past, 15, 153–168, https://doi.org/10.5194/cp-15-153-2019, 2019.