

Chapter 2

ESM Workflow

V. Balaji and Amy Langenhorst

The period 2000–2010 may be considered the decade when Earth system modeling (ESM) came of age. Systematic ESM-based international scientific campaigns, such as the Intergovernmental Panel of Climate Change (IPCC), are recognized as central elements both in scientific research to understand the workings of the climate system, as well as to provide reasoned and fact-based guidance to global governance systems, on how to deal with the planetary scale challenge of climate change.

The running and analysis of ESMs is a huge technical challenge, whose several elements have been discussed in previous volumes. The problems range from assembling models from across a community of specialists in different aspects of the climate—oceanographers and stratospheric chemists, for example—to executing climate models on complex supercomputing hardware, to developing global distributed data archives for coordinated international experiments.

At the modeling centers, we are experiencing a serious escalation in the demands placed on us. There is an expectation that ESMs will become “operational”: used as forecast tools on time scales of seasons to decades; used to generate scenarios of future climate change in response to particular policy choices; used as a planning tool in industries from energy to insurance.

This trend implies a radical shift in the way we do modeling: *an integrated infrastructure for the building and running of ESMs and analysis of ESM output data*. What is now a heroic effort involving a motley crew of scientists, software engineers and systems analysts has to become a streamlined process: a *scientific workflow*.

V. Balaji (✉)
Princeton University, Princeton, USA
e-mail: balaji@princeton.edu

Amy Langenhorst
High Performance Technology Inc, Princeton, USA
e-mail: amy.langenhorst@noaa.gov

2.1 What is a Workflow?

A *workflow* is a more or less formal description of a sequence of activities. The description is intended to be precise and comprehensive. In the context of science, it provides *reproducibility* of an experiment. For sciences such as Earth system modeling that are based on digital simulation, this can in principle allow *exact binary reproducibility* of an experiment.

Once a workflow is fully described, it can not only be reproduced exactly; it then becomes easy to define perturbations on those experiments as well, for instance by rerunning the experiment varying a single parameter. Two such perturbed experiments can be compared by differencing the workflows: every element in the workflow will be identical save the one parameter that was varied.

Finally, the formal description serves as input to a *curator* of model output. Since all of the scientific, numerical and technical input into the model is specified in the workflow, it serves as a complete descriptor of the model output as well.

In the next section we provide an example of a functioning workflow system, and use it to develop a general architecture for workflows and curators in the context of Earth system modeling.

2.2 Example: FRE, the FMS Runtime Environment

Weather centres and operational sites have incorporated workflows for running their models [e.g. *preplFS* developed at European Centre for Medium-Range Weather Forecasts (ECMWF) and in use at several weather centres Larsson and Wedi (2006)] but those tend to have restricted palettes of choices, more attuned to operational uses. In this section, we describe in some detail a scientific workflow for ESMs aimed at development and testing as well as operational running of models, as an example to illustrate the range of issues one must deal with in workflow development. FRE, the FMS Runtime Environment is a workflow system developed for the Flexible Modeling System (FMS) described in Volume 3 of this series. FRE has been an operational system for models at the Geophysical Fluid Dynamics Laboratory (GFDL) since 2002, and was a critical element in the process of running GFDL models for IPCC AR4 in 2003–2004, and ever since.

FRE was developed with the goal of simplifying the process of assembling, running and analyzing FMS-based models. Steps in the modeling process include:

- *retrieve known site configuration information.* For instance, best-practice compiler and library configurations, paths to needed standard and bespoke utilities for data processing; instructions to a batch-processing system.
- *retrieve known component configurations in a multi-component model.* As in many ESMs that follow component-based design, an FMS model is composed of many components. Within FRE, an oceanographer would be able to focus on maintaining exquisite control of her own component, while inheriting an atmosphere component's configuration wholesale from another experiment.

- *perform basic tests before launching a run.* FMS best practice requires running some standard tests before launching. A basic compatibility check tests that the ESM can be run forward stably and safely for short duration. Further, since we know that an ESM run involves a long series of runs, we must verify that the model saves state (“restart files”) with adequate information. A simple test of this is to run the model for 2 days in a single run, and then again in two segments of 1 day. At the end, both runs should provide bitwise-identical results. A second test might be to see if the model provides the same answer at different processor counts. Another form of basic test might be to check the integrity of input data files which might have been acquired from a remote repository, by verifying their checksums.
- *launch, manage and monitor a long-running job sequence.* A single job from an ESM simulation submitted to a batch system is usually a small fraction of the entire run length. FRE jobs are designed to resubmit themselves until a specified end time, or to run indefinitely until manually interrupted. A user is able to query the model state.
- *launch, manage and monitor post-processing, analysis and data publication.* Post-processing (see Vol. 4) is itself an intensive task rivalling in resource consumption the model run itself. This step can take us all the way to the preparing of graphical output for a scientific audience, and publication of data on a public portal.

An experimental configuration is expressed in an XML (Bray et al. 2000) file called a “FRE file”. The Extensible Markup Language (XML) is a common choice for such applications for various reasons: it is textual, easy to manipulate and edit, with a rich and readily-available environment of tools and software.

A single FRE file contains the complete workflow of an FMS experiment. Thus at any stage, any aspect of the experiment may be queried: a user looking at graphical output from an experiment might ask for the value of some input parameter, or what tracer advection scheme was used by the ocean, or how many processors were used by the job; all of this information is at her fingertips, in the FRE file.

The various stages in the workflow are each executed by a script, which may be interactive on the desktop, or remotely executed on a batch system. The scripts generated by FRE are generally in the C-shell (csh). These shells scripts themselves are generated by command-line¹ “meta-scripts” that process and interpret the FRE file to generate the requisite instructions: for instance `fremake` generates compilation instructions for the experiment; `frerun` generates the runscript to execute the experiment. These “meta-scripts”, known as FRE scripts, are written in languages like perl and python.

A fragment of a FRE file is shown here for discussion:

The XML fragment above illustrates some key aspects of FRE workflow.

- The basic workflow unit of FRE is an `<experiment>`.

¹ FRE at the moment of writing is entirely command-line based: a graphical user interface is conceivable, but has yet to be built. We note that different labs have cultural biases toward the command line or the GUI: GFDL appears to be of the textual rather than iconographic culture.

```

<experiment name="CM2.4C_U1" inherit="CM2.4C.base">
  <component name="fms">
    <source vc="cvs" root="/home/fms/cvs">
      <codeBase version="perth">shared</codeBase>
      <csh>
        cvs up -r perth_bw time_interp.F90
      </csh>
    </source>
    <compile>
      <cppDefs>-Duse_libMPI -Duse_netCDF</cppDefs>
    </compile>
  </component>
  <component name="mom4p1" requires="fms">
    <compile>
      <cppDefs>"-DUSE_OCEAN_BGC"</cppDefs>
    </compile>
    <compile target="static" >
      <cppDefs>"-DUSE_OCEAN_BGC
        -DNI=1440 -DNJ=1070 -DNK=50"</cppDefs>
    </compile>
  </component>
</experiment>

```

- *Inheritance* is a key concept in FRE: an experiment can inherit all of another experiment, and override only those aspects in which it wishes to differentiate itself. This lends itself to an extremely compact description of a series of related experiments, each of which represents a perturbation with respect to some base experiment. One of the FRE utilities, *freCanon*, can convert the compact form to what is known as the canonical form, in which all inherited values are resolved and independently expressed.
- Within an experiment, information is organized by `<component>`. Here we see a component called `fms`, which is the FMS infrastructure itself, and another component called `mom4p1`, which represents the Modular Ocean Model Version 4 (MOM4) ocean model. For each component, we can have several *methods*, each representing a different stage in the workflow. Typical stages in the workflow include `<source>`, which prepares the source code; `<compile>`, which prepares the executable. Other stages in FRE workflow include `<input>`, which prepares input data; `<run>` and `<postProcess>`, not shown in the XML fragment above.
- `<source>`: each source can come from its own repository. The `vc` attribute indicates the type of version control, e.g. `cvs` or `Subversion`, and `root` the path to

its associated repository. The `<cs>` tag can exist for any method, where the user can indicate specialized processing not provided by the standard FRE utilities.

- `<compile>`: this method provides instructions for compilation. There can be multiple compilation *targets*: each target has different compilation instructions, and thus represents a different realization of an experiment. There can be dependencies among components in compilation, indicated by the `requires` attribute. The dependency indicates compilation order, for example: the `fms` component may need to be compiled first.²
- `<input>` (not shown in Code block 1): lists input parameters and datasets for a component. Input parameters are organized into `<parameterGroup>` s which may be converted by FRE utilities into Fortran namelists, for example. Input datasets follow this syntax:

```
<dataFile source=.. version=.. timestamp=.. checksum=..>
  INPUT/aerosol.nc
</dataFile>
```

As noted here, datasets themselves are under version control. The dataset is delivered to a specific target `INPUT/aerosol.nc` where it will be read by the model code.

The source could be under hierarchical storage management (see Vol. 4, Chap. 3, Sect. 3 for instance) and involve specialized instructions for retrieval from “deep storage”. These instructions are specified in the platform and site configuration sections of FRE, not shown here.

An alternate method of retrieval might be from some remote resource, following the OPeNDAP protocol (Cornillon et al. 1993), for example.

Datasets themselves might be under version control, indicated by the optional `version` attribute.

Finally, the `timestamp` and `checksum` attributes are used to verify data integrity of input datasets.

- `<run>`: FMS typically runs in single-executable (SPMD³) mode. Thus only one component, the top-level `coupler` component, has an associated `<run>` method. However, it is easy to express MPMD programming in FRE should one choose to, by assigning `<run>` methods to multiple components. Runs are specified in `<segment>` s, each of which constitutes one job on the system. A full model run may be centuries long in model time, however a single run segment is constrained by the limits on the scheduler, how long it permits a single job to remain on the system. Run segments save their states and then resubmit themselves until the end of the simulation. *Production* runs also typically save a

² FMS is written in F90, where compilation order matters.

³ SPMD, or single-program multiple-data compiles a multi-component model into a single executable. Other modeling systems, e.g. PRISM, employ the multiple-program, multiple-data (MPMD) mode.

lot of history data and launch a parallel series of post-processing jobs, described below; *development* or *testing* runs save minimal data for immediate analysis. Runs intended for regression testing also list a <reference> run, with which they are expected to match answers.

- **<postProcess>**: this method refers to data processing performed upon the output of run or run segment. For example, one might wish to regrid output from the model's native grid onto some standard spatial grid; compute statistics; derive auxiliary variables from saved variables. These computations are data- and I/O-intensive, and do not readily lend themselves to parallelism (see Volume 4 for an in-depth discussion of parallel I/O). Parallelism is at the script level, where processing steps with no mutual dependencies are scheduled concurrently. FMS diagnostic output ("history") is organized into subcomponents by time or space sampling. Under **<postProcess>** for the **atmos** component, one might find subcomponents for monthly and daily data, for instance. Subcomponents may also have associated **<analysis>** methods. Once the post-processing is complete, the data is then processed by visualization and analysis engines to generate graphical output for the user. The ability to run a whole suite of standard analyses on any model run is one of the most compelling features of FRE, the proof of its claim to be an "end-to-end" solution.

This summary look at the structure of a FRE experiment, its components and their methods, provides a glimpse into a functioning and mature *ESM workflow*. Its versatility is shown by the variety of command-line FRE utilities that work off the FRE file:

fremake prepare source and executables for an experiment.

frerun launch and manage a FRE run, in production, development and test modes.

frecheck compare a test run against a reference run, and report discrepancies.

frestatus monitor and report the state of a running FRE experiment.

frepp FRE post-processing and analysis.

freppcheck monitor a post-processing job sequence and report discrepancies and missing data.

frelist list experiments associated with a FRE file, including their inheritance structure.

fre canon convert an experiment expressed with inheritance into a solo FRE experiment with no dependencies.

fredb enter an experiment into the GFDL Curator, a **mySQL** database of model experiments. This powerful backend system underpins FRE's ability to retrieve prior model configurations.

freversion The FRE syntax evolves as FMS adds new features and components. This utility provides a translator between versions of FRE syntax, and allows users to update their FRE files.

We next discuss lessons learned from FRE, and how workflow concepts might inform the field of Earth system modeling.

2.3 Discussion: Workflows and Curators

Just as FMS provided inspiration and testing grounds for concepts of modeling frameworks that informed the design of ESM frameworks like ESMF and PRISM, so too did FRE lead to efforts to standardize some of these ideas over a larger community, in the Earth System Curator (ESC) project (Dunlap et al. 2008). That project principally followed from FRE's insight that the workflow constitutes the best available description of a model, and thus its *provenance metadata*. The ESC project has had considerable success in specifying formal descriptions of Earth system models constructed from components, and in conjunction with its European counterpart METAFOR (<http://metaforclimate.eu>), is developing provenance metadata for models participating in large coordinated international experiments, such as the IPCC AR5, scheduled for 2013. By capturing workflow information in a curator database, we will be able to provide some of FRE's services for a wider community.

The architecture of an ESM workflow can also be specified in generic form as in Fig. 2.1. And generic specification systems for scientific workflow such as Kepler (Ludascher et al. 2006) are now becoming popular. Is the time now ripe for an effort to build a standard ESM workflow system? It is tempting to think that a system built on top of a standard framework could indeed implement such a workflow. One could go further and even conceive of model components becoming available as web services that can be directly interfaced from other applications.

While the ESC project does indeed contemplate a Curator Runtime Environment (CRE), there are some caveats to be borne in mind. Principal among these is what is known as the “fuzzy boundary problem” (Sessions 2004–2005). This highlights the fact that the same thing might appear as an *object* (internal data structure) in one application, as a model component in another, and as a web service in a third. In the context of ESMs, we could for instance imagine $g=9.81$ being a line of code in one model, the value of g being read from a binary file in another model component, and being part of the workflow specification in a third. It is imperative to maintain this degree of flexibility for the developer, and not unduly constrain her in her methods of specification. It is in fact very common indeed for the same application to evolve toward greater formality over time, but that must occur at its own pace.

Nonetheless, the benefits of a more or less formal expression of workflow, and its incorporation into metadata seem to be evident, and we foresee it becoming more prominent in our field with time. Not only does this seem a necessary step in the “operational” use of ESMs alluded to in the beginning of this chapter, there are many new applications that are only dimly seen now, that will become possible with the adoption of formal workflows. For instance, current scientific publications based on model results provide, at best, a reference to the model output data on a server somewhere. We can imagine instead that a scientific article could one day contain a link or URL pointing to a complete workflow file attached to some hardware resource, say a computing “cloud”. A scholar reading that article would be able then to manipulate that model directly through the workflow specification, for instance

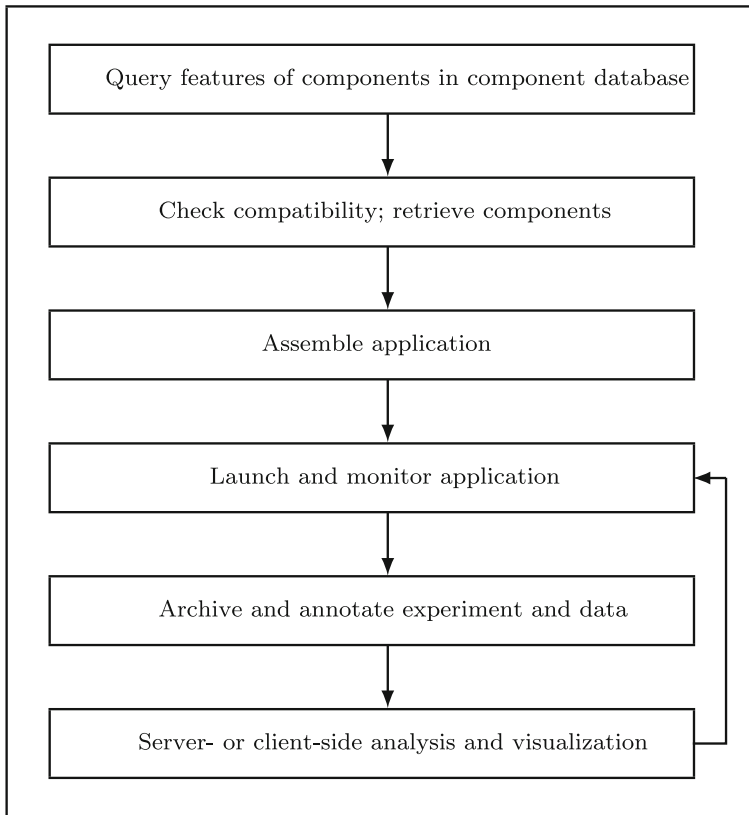


Fig. 2.1 Generic ESM workflow

by rerunning that model varying some parameter! Such an application of workflows would be transformative not only of the field, but of the nature of scientific publication itself.

References

- Bray T, Paoli J, Sperberg-McQueen C, Maler E, Yergeau F (2000) Extensible Markup Language (XML) 1.0. W3C Recommendation 6
- Cornillon P, G F, J G, G M (1993) Report on the first workshop for the distributed oceanographic data system. Technical report, Graduate School of Oceanography, University of Rhode Island
- Dunlap R, Mark L, Rugaber S, Balaji V, Chastang J, Cinquini L, Middleton D, Murphy S (2008) Earth system curator: metadata infrastructure for climate modeling. *Earth Sci Inform* 1(3–4): 131–149
- Larsson C, Wedi N (2006) prepIFS, a software infrastructure tool for climate research in Europe. *Geophys Res Abstr* 8:04250

Ludascher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, Lee E, Tao J, Zhao Y (2006) Scientific workflow management and the Kepler system. *Concurr Comput* 18(10):1039

Sessions R (2004–2005) Fuzzy boundaries: objects, components, and web services. *ACM Queue* 2(9):40–47

Earth System Modelling - Volume 5

Tools for Configuring, Building and Running Models

Ford, R.; Riley, G.; Budich, R.; Redler, R.

2012, XV, 97 p. 23 illus., Softcover

ISBN: 978-3-642-23931-1