



# Configuration Management Best Practice guide [Draft]

---

## *Temporary explanation (to be removed from the final document)*

*IS-ENES2 is a European Union funded project on technical infrastructure for climate science.*

*As part of the IS-ENES2 Mid-Term review, the EU project Officer asked us to provide community recommendations on Configuration Management (CM). This document provides these community recommendations.*

*This document is being developed as part of a “virtual workshop” and the aim is to get a consensus across as many institutions as possible. The contributors and reviewers will put their names to this document on behalf of their institution.*

*We need to be clear that there is no funding within IS-ENES2 to implement any such recommendations, and so this is not a commitment to adhere to them. But we are looking for agreement on what we think the community should do, given time and resources.*



## Authors and Reviewers

This document has been written to provide advice for the climate science community on configuration management with the aim to be authored for the community and by the community. Much of this advice is also applicable to the weather science community. The document has been coordinated by Mick Carter in a virtual workshop as part of the IS-ENES2 project. This project has received funding from the EU's Seventh Framework Programme for Research (grant agreement number 312979). The document has been jointly developed with input, comment and/or review from the following people representing their institutions:

Ag Stephens: STFC CEDA (Science and Technology Facilities Council)  
Alf Grini: Met.No (Meteorologisk Institut)  
Amy Langenhorst: NOAA (National Oceanographic and Atmospheric Administration)  
Antonio S. Cofino: UC (Universidad de Cantabria)  
Arnaud Caubel and Marie-Alice Foujols: CNRS-IPSL (Centre National de la Recherche Scientifique)  
Christoph Heinze: UiB (Universitetet i Bergen) )  
Camiel Severijns: KNMI (Koninklijk Nederlands Meteorologisch Instituut)  
Colin Jones NCAS/MetO (UK Earth system modelling project, National Centre for Atmospheric Science and Met Office)  
Claire Levy: NEMO – LOCEAN (Laboratoire d'Océanographie et du Climat)  
Giovanni Aloisio & Silvia Mocavero: CMCC (Centro Euro-Mediterraneo sui Cambiamenti Climatici)  
Glenn Carver & Paul Burton: ECMWF (European Centre for Medium-Range Weather Forecasts)  
Graham Riley: UNIMAN (The University of Manchester),  
Grenville Lister: UREAD-NCAS (The University of Reading - National Centre for Atmospheric Science)  
Hamish Struthers: LiU (Linköpings Universitet)  
Johan Lee & Yoonjae Kim: KMA (Korea Meteorological Administration)  
Kerstin Fieg: DKRZ (Deutsches Klimarechenzentrum, GMBH)  
Laurent Chardon: ECCC (Environment and Climate Change Canada)  
Mariana Vertenstein, NCAR (National Center for Atmospheric Research)  
Marius Matriata: INHGA (Institutul National de Hidrologie si Gospodarie a Apelor)  
Mick Carter & Dave Matthews: MetO (Met Office)  
Michael Naughton: BoM (Bureau of Meteorology)  
Michael Uddstrom & Hilary Oliver: NIWA (National Institute of Water and Atmospheric Research)  
Neil Chue Hong & Simon Hettrick: SSI (The Software Sustainability Institute)  
Ole Bøssing Christensen: DMI (Danmarks Meteorologiske Institut)  
Oriol Mula Valls & Domingo Manubens IC3/BSC (Institut Català de Ciències del Clima)  
Reinhard Budich, Luis Kornblueh & Karl-Hermann Wieners: MPG (Max-Planck-Gesellschaft zur Förderung der Wissenschaften)  
Scott Wales: ARC (Australian Research Council Centre of Excellence for Climate System Science)  
Sophie Valcke: CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique)  
Stéphane Sénési: MF-CNRM: Meteo-France  
Uwe Fladrich & Martin Evaldsson: SHMI (Sveriges Meteorologiska och Hydrologiska Institut)  
V. Balaji: GFDL (Geophysical Fluid Dynamics Laboratory)  
Veronika Eyring & Axel Lauer: DLR (Deutsches Zentrum für Luft- und Raumfahrt)



## Introduction

Configuration management (CM) is defined as a “process for establishing and maintaining consistency of a product's performance, functional and physical attributes with its requirements, design and operational information throughout its life<sup>1</sup>.”

Good configuration management is very important for the climate community primarily because:

1. It is key to maintain the ability to reproduce results, which is central to the scientific method;
2. It increases scientific openness by supporting clarity of what has been done;
3. It increases the efficiency of the scientists when developing, debugging and dealing with failures on the computers they run on;
4. It provides a defence against those who aim to discredit climate science;
5. It allows working practices to scale beyond small groups and improving the ability of the climate community to work collaboratively within and across institutions;
6. It improves quality and reduces risk due to human error;
7. It provides the level of rigour required to allow climate science to provide operational services, such as seasonal predictions and hence supporting the Climate Services agenda.

There are four key levels for CM within climate modelling science:

1. the Software CM<sup>2</sup> of the model codes such as an atmospheric, oceanic or sea-ice model. This includes software version control of the code base and the QA processes that manages that code base;
2. the CM of the model configurations – the choices of components with options and parameters, the resolution and domain etc.;
3. the CM of the full workflow<sup>3</sup> required to complete climate simulations and to provide standardised output, including compiler options. This includes any processes that run before, after or alongside the models to prepared and or deal with the model inputs and outputs.
4. the CM of the full workflow required to analyse and evaluate the standardised output from climate model simulations with community-tools.

In its entirety, CM needs to

1. Evaluate proposed changes and plan change activity leading to a release of a reference version of the model codes, configurations and workflows;
2. Track the activity delivering those changes in a way that ensures appropriate quality control (including testing and review);
3. Identify specific reference versions of model codes, configuration and workflows for release (sometimes called version control).

---

<sup>1</sup> See [https://en.wikipedia.org/wiki/Configuration\\_management](https://en.wikipedia.org/wiki/Configuration_management)

<sup>2</sup> See [https://en.wikipedia.org/wiki/Software\\_configuration\\_management](https://en.wikipedia.org/wiki/Software_configuration_management)

<sup>3</sup> <https://en.wikipedia.org/wiki/Workflow>



The difference between versions of the model codes, configurations and workflows is something the community needs to be clear about and where different terminology can confuse. The model codes that are developed can support many configurations: different resolutions, different domains (global, regional, etc), different components and schemes and different settings of controlling parameters and input data files for example. Both the model codes and the configurations need to be properly version controlled using CM principles. A particular version of a model code (or more typically a set of versions of model codes for complex earth system models) can support many configurations and a particular configuration may be available in more than one model code version. It is advisable to have distinct naming conventions to distinguish these things and to avoid confusion. A workflow will include one or more model configuration and other processes, such as pre and post processing of data. Because a workflow encompasses model configurations, the remainder of this document will focus on the need to configuration management of model codes and workflows but there are advantages in separating the definitions of the configuration of models and the workflow. This separation allows models to be more easily used across workflow solutions.

## Analysis

The configuration management workshop provided evidence from the view of a number of different actors or roles. From these best practice guidelines have been derived and are proposed. The analysis and advice has been broken down into the needs of these various actors.

### The model or component developer

It is standard working practice for software developers to use a version control tool, such as Subversion or Git, to help manage their development. Such tools provide a convenient way to be efficient, giving access to any committed version of the code integrated with record keeping to record notes about the changes and the ability to revert changes. The tools are able to support a wide range of working practices, but it is important that a team of developers working on the same code have processes that lead to a single, central repository for their project<sup>4</sup>, owned and curated by the owner of the model or software system being developed, with a common configuration working practice and version naming conventions. Further, teams benefit from being able to see code from their colleagues on branches. With the wide availability of globally accessible code repositories, such as Github, it is clear that distributed development for a central, single repository is an achievable aim for the community for any one code base. It has also been shown to be possible for a project

---

<sup>4</sup> There is no proposal here to share a repository across codes bases or institutions. Users of Subversion will often have a number of related projects within one repository where-as users of Git are likely to have a more modular repository design.



to use a single global repository using Subversion that supports international development<sup>5</sup> but the Git support for local repositories has some advantages<sup>6</sup>.

Working practices need to be clearly defined and suitable education needs to be given. People need to know how to gain access to the repositories and the rules and controls for putting changes onto the trunk. A common approach will help people who need to work across multiple projects or organisations and when they move between them. Scientists are also developers within the climate modelling community but often come from cultures that do not use such tools and so the proposed guidance and training is particularly important for climate scientists. Further, working practises need to cater for a significant amount of parallel development because many scientists are often involved and work to varying timetables, depending on the maturity of the science they are developing. Development needs to be done as close as possible to the head of the trunk of the CM system to reduce the, often significant, overhead of resolving clashes.

Various levels of formalism are required, ranging from less formal research and development activities, such as developing a new scientific formulation for comparison with established codes through to formal development of codes for use within production or operational activities such as CMIP exercises or seasonal prediction systems.

The climate scientist also needs to apply good change-control discipline as widely as possible to all the code they develop, including personal code for things like data processing.

All the developments should be well-documented to promote easier maintenance and the sharing of software within the community.

It is inappropriate to advise on a single version control tool for the whole community as the difference in benefits between the modern tools (based on Git or Subversion) are small and the cost of moving an institution or project to a new tool is high.

In summary, the recommendations are to ensure as much code as possible is developed within a configuration management tool using well-documented, common working practices and an appropriate level of formalism. There should be training and support to achieve this, recognising that not all developers have a background in software engineering. There is a good deal of open source material that can form the basis of such training. For example, Software Carpentry's<sup>7</sup> basic courses on code management and version control have been successfully adapted for delivery in many different disciplines, including the climate sciences.

---

<sup>5</sup> The Met Office has provided a repository service using Subversion that is used by the Unified Model Partnership: <http://www.metoffice.gov.uk/research/collaboration/um-collaboration>. The Earth System Model Evaluation Tool (ESMValTool, <http://www.esmvaltool.org/>) is developed in a Subversion controlled repository with regular releases as open-source software for use by the wider community.

<sup>6</sup> Git's cloning and synchronising (push) functionality is described at <http://gitref.org/>.

<sup>7</sup> <http://software-carpentry.org/>



## The owner of a model or system

The owner of a model or system needs to ensure that control is provided on the content of new releases of a set of software defining the model or system. The owner also needs to ensure that quality is maintained in that set of software. In this community, software is developed by a very wide range of people who sit within a broad management structure. For example, scientists not under the control of the owner will be developing the code within a model. Further, people may start a development with the purpose of trying out a new idea which can be considered a “one shot” development, but care needs to be taken to ensure that the code quality is improved to an appropriate level before it is included in a model release to ensure that the code is sustainable.

Project management tools that allow direct, clickable, links from issues to the relevant code changes are highly recommended. Examples of systems that can do this include JIRA, Redmine and Trac, which provide ticketing or issue tracking and are integrated with configuration management tools. The model and system owner needs to:

- plan what changes can go into a release to provide an achievable and consistent aim;
- ensure the assignment of tasks, taking into account the expertise of each developer;
- ensure design review and code review;
- ensure both regression testing of existing functionality the testing of new functionality; and
- ensure integration of a change into the wider system.

An example of the need for consistency in a release is a technical development that may be done by scientific software engineers that is required to support new science developed by a scientist.

All this can be controlled and monitored through a suitably configured system built upon services or tools such as JIRA, Redmine or Trac, all of which can be configured to cover the needs of planning, management and quality assurance. These tools have all been used to good effect within the community.

Typically, the owner will control the trunk of a code repository from where releases are made. The model developer will work on branches for research and development purposes only merging code onto the trunk under control of the system manager’s processes and following the necessary quality assurance processes such as review and testing. The most effective processes will promote a more continuous integration of changes onto the trunk and support regular, automated testing.<sup>8</sup>

As well as coding quality and regression testing, there is also an important management role in ensuring that new model science has been properly assessed. The process of scientific validation of climate models is complex, and not the focus of this paper, but care needs to

---

<sup>8</sup> [https://en.wikipedia.org/wiki/Continuous\\_integration](https://en.wikipedia.org/wiki/Continuous_integration)



be taken to ensure there is a clear policy on how codes are managed that are complete in themselves but not yet part of a released model configuration (which will often combine a number of scientific changes). Some models are able to support scientific developments as options onto the trunk before they have been fully accepted into in a production model configuration. In this case, the regression testing of existing formulation is essential. Appendix 2 discusses model evaluation.

The owner of a model or software system needs to devise clear naming conventions for releases<sup>9</sup> and an organisation needs to ensure distinct naming conversions between model codes and model configurations.

In summary, the owner of a model or other software system needs to maintain the quality of the code base by ensuring that appropriate practices are defined and applied (design, review, testing, documentation and consistency). A tool should be selected that can monitor and help control the development work and help support the processes in an efficient way. They need to ensure that repositories have a clear structure. For model configurations, there is also the important task of assessing and controlling the improvement of the science in the model.

## The experiment designer and manager

Doing experiments using climate models is increasingly complex. Experiments need to couple multiple models, often developed by different institutions. Input data needs to be controlled. Experiments are increasingly made up of an ensemble of integrations. Data needs to be post-processed in a controlled, systematic way.

Increasingly, workflow tools are being used to manage the complexity of experiments where scripts have been used before. It is important that suite definitions, that each describe the implementation of a specific workflow, can be shared and adapted to the needs of a particular experiment whilst maintaining their provenance and that it is possible to know exactly what was run from end-to-end to provide reproducibility.

To make the most of the benefits of both workflow tools and configuration management, a number of institutions have developed frameworks that provide the ability to deliver configuration managed (i.e., reproducible) experiment design to complement the configuration management of the scientific codes themselves. Another issue that is increasingly complex for people running experiments is that of managing the provenance of external libraries used by the models. To ensure results can be reproduced, libraries need to be built from well defined source codes, uniquely named and a safe way of selecting the right libraries and recording what has been used needs to be developed. This is further complicated by the fact that different models used within a coupled earth system model may rely on different versions of the same library. That latter challenge is one for the build process and workflow, rather than a configuration management system, but these systems

---

<sup>9</sup> See <http://semver.org/> for standard flexible naming convention that can be a starting point for this exercise.



need to ensure that enough information is recorded to allow results to be reproduced in the future<sup>10</sup>. See Appendix 1 for a more in-depth discussion on reproducibility in climate models.

Examples of systems that apply CM to workflows in this community include:

1. The ECCC uses a framework called Maestro that handles execution traces, error reporting, deployment to platforms, user inputs, experiment content and configuration as well as their workflow.
2. The system used at ECMWF combines prepIFS which records experiment configuration options with standard version control tools for source code, scripts and ecFlow based suite definitions.
3. The FRE Curator system used at GFDL. This is a meta-data centric solution that unifies access to climate modelling components and output datasets through a central meta-data repository. See <http://www.gfdl.noaa.gov/modeling-systems-group-fre-curator> .
4. Autosubmit (<http://www.bsc.es/earthscience/autosubmit/>) the solution developed at IC3 and BSC to run and manage experiments. Autosubmit contains information about the model version, the experiment configuration, the workflow to be run and the computing facilities used.
5. The libIGCM ( <http://forge.ipsl.jussieu.fr/libigcm> ) running environment developed at IPSL is based on a portable, modular, efficient library of ksh-functions and ksh-scripts. libIGCM is managed and shared through a Subversion repository.
6. The Met Office's Rose system that has suite definitions kept in a subversion repository that can version control the configuration of all the applications in an experiment and the Cylc-managed workflows of those applications. See <http://metomi.github.io/rose/doc/rose.html>.
7. The compset concept used at NCAR for the CESM model. This uses a naming convention to define a specific combination of components and their configuration. See [http://www.cesm.ucar.edu/models/cesm1.0/cesm/cesm\\_doc\\_1\\_0\\_4/x42.html](http://www.cesm.ucar.edu/models/cesm1.0/cesm/cesm_doc_1_0_4/x42.html) .

In summary, the experiment designer needs to ensure that the whole workflow can be reliably repeated and the use of workflow management tools that support CM is encouraged to achieve this.

## The consumer of the output from experiments

The user of the output of an experiment needs to have access to definitions of the experiment design and configuration in varying levels of detail and they need this information in human readable form. This includes detailed information on the model output such as exact variable definition, units, averaging intervals, etc. This requirement contrasts with the core need of the experiment designer who primarily needs to manage the

---

<sup>10</sup> The XALT and EasyBuild are tools that are worth investigating to help with this problem. See [http://sc15.supercomputing.org/sites/all/themes/SC15images/tech\\_poster/tech\\_poster\\_pages/post281.html](http://sc15.supercomputing.org/sites/all/themes/SC15images/tech_poster/tech_poster_pages/post281.html) and <https://hpcugent.github.io/easybuild/>.



configuration of machine readable input data used by the tasks in the workflow rather than providing information to end users.

The generation of meta-data for the end user of the data from an experiment is a very time-consuming task. To reduce this overhead, it is important to automate the gathering of meta-data from the workflow tool and to present the experiment designer with an interface that is as simple as possible to both design and run the experiment in the repeatable way and to provide the necessary meta-data to the end-user. Hence, automated meta-data capture should be integrated with workflow tools where possible.

However, complete automation is not always possible, especially at the time of data production. For example citation meta-data for an experiment refers to the published papers and contains such things as author(s), year, title, publisher, reference (if any). These are seldom finalised when experiments are run and so a manual element to meta-data gathering will always be necessary but the burden can be reduced by automation.

In summary, the consumer of the data from climate integration relies on good CM being applied in all the roles reading to the workflow that produced the model output they are using. They are also reliant on having access to descriptions of the models and other systems that produce the results so that they know how to use it.

## Summary & Conclusions

Individual institutions already use a number of version control tools that support various aspects of configuration management. As well as the recommendations provided for each of the roles described above, this paper recommends that the community should move to extend the scope of their existing configuration management infrastructure to also include, as far as possible:

1. the complete workflow from ancillary file generation through to final results, including the model configurations and model codes;
2. compiler and link information, including the libraries that the model codes rely on to allow results to be reproduced; and
3. tools and processes to support the management of the code base to improve code quality by tracking testing and review processes.

All these extensions should be designed to capture the information required to make the experiments and standard analysis reproducible. As partner sites already have working CM systems and processes, it is not appropriate to recommend that people with existing solutions should move to any particular system, given the cost of such a move and the, often, modest benefit it would bring. However, the development of this document has demonstrated the benefit of sharing experiences and best practice.<sup>11</sup>

---

<sup>11</sup> The IS-ENES2 project coordinated an independent evaluation of the FCM system for the community with work done at ECMWF, IPSL and MPG. FCM provides an alternative interface to Subversion which supports a particular way of working, making it easier to work with branches, for instance. It is also integrated with a build system (FCM make) which helps maintain the repeatability of the build process as it builds directly from code



## Appendix 1. Climate Model Reproducibility

There are two types of reproducibility that concern the climate scientists: scientific reproducibility and bit-wise reproducibility. The latter emerges because exactly the same code, when correctly executed on different platforms or with different compilers, will generate different results which originate from such things as the different order of operations that can occur when compilers optimise codes for a particular CPU. Climate models are simulating chaotic systems and so tiny differences often grow quickly and will change the evolution of a climate experiment. Climate scientists develop analysis techniques<sup>12</sup> that eliminate this variability so that the same scientific conclusions are drawn from scientifically identical models that are run on different platforms giving different evolutions. In some circumstances, there are benefits in being able to recreate exactly the same results at the bit level, for example to recreate some data from the middle of a climate integration that may have been lost or to extract further detail on a particular event within an integration. Not all climate models aim to provide this functionality or only provide it when using the same parallel domain decomposition. To achieve bit-level reproducibility, the experimenter needs to know:

- the exact code that has been run,
- the input data and controlling parameters to all tasks in a workflow,
- the build process parameters, including compiler flags,
- any libraries that have been used that could impact the results at the bit level
- the compiler used,
- the architecture of the hardware the model has been run on.

And they also need access to all those things. To facilitate this, compilers and libraries need to be well managed, referenced (using unix “modules”, for example) and then recorded – preferably automatically. How this is achieved will depend on the available tools and systems used at a site. Because such information is only useful to people who have access to the same resources and only for the period in which they do have access, ad-hoc methods can be sufficient in certain circumstances but given the increasing complexity, automatically capturing this information is increasingly important.

It is even more important to ensure that scientific reproducibility is well governed. That is, climate scientists need to be able to repeat an integration that is scientifically the same, even though the evolution diverges as a result of the low-order bit differences that occur when using different chip architectures, compilers etc. This requires the proper configuration management of a subset of the items listed above:

---

stored in Subversion and combines together code from multiple repositories when necessary. Although not strictly configuration management, the FCM make system was evaluated as a powerful tool for dealing with fortran code dependencies and parallelising the build process. Further information can be found on the IS-ENES2 web site

<sup>12</sup> Typically statistical techniques that look at trends or changes in frequency of events or use ensemble methods to eliminate the variability.



- the exact code that has been run,
- the build process parameters, where these change order of calculation in a significant way,
- The input data and controlling parameters,
- any libraries that have been used that could impact the results at the scientific level (e.g. the accuracy of solvers provided by libraries).

Again, the workflow tool will often define input data and parameters. It is also important to consider the configuration management of all inputs, be they libraries controlling parameters or input data, such as ancillary files. Ancillary file generation has its own, sometimes complex, workflow starting with observational data sources followed by creation of gridded datasets that then need to be made consistent with other gridded datasets, such as land-sea masks and finally re-gridding onto the model's grid. There are scientific choices to be made at each stage and these needs to be recorded alongside the configuration management of the full workflow.



## Appendix 2. Climate Model Evaluation

Climate model evaluation is a complex topic and a key part of the quality assurance process for both model codes and model configurations. The details of this endeavour are beyond the scope of this document, other than to recognise its importance.

The person who owns a model configuration will need to satisfy a number of customers in ensuring that the model configuration is fit for the purposes for which it is going to be used<sup>13</sup>. No climate model is perfect and there will be balances to be made in terms of the model's performance between different aspects, such as its cost and its ability to represent the individual processes modelled and its performance in representing current climate and past climate change. The quality of a configuration will depend on many things like model resolution, quality of ancillary data and the quality of the formulation of the various schemes. It is most effective to have well defined evaluation criteria set for the task of developing model configurations and these criteria tend to be improved over time as a result of experience or a desire to measure new aspects as a result of increased fidelity or the addition of new processes in the models. Also, a set of test cases that explore various aspects the model is also useful.

There are occasions where the scientific results of a given configuration need to be validated against a previous result even when there have been no scientific changes. The cause of this is explained in Appendix 1 - small numerical differences are caused by changes in the order of calculation, machine differences and compiler differences. These differences grow quickly in climate models and this divergence of solution can potentially hide bugs. Then following techniques can help in such cases:

- Checking a model gives the same results technically, for example on different processor decompositions and across model restarts. This can eliminate technical errors.
- Checking that a model's rate of divergence matches the rate of divergence measured when perturbing initial conditions by a tiny amount over a short time period. A reference model is run a number of times with perturbed initial conditions and the test model's divergence is checked against this set.<sup>14</sup>
- Unit tests of individual schemes within a model can help spot bugs if well designed and this can avoid the complexity of tracing errors back to their source in complex climate models.
- Longer integrations should be run to show that the model's climatology has not changed significantly. The length of the run needs to be long enough to eliminate the impact of natural variation.

---

<sup>13</sup> <http://www.easterbrook.ca/steve/2010/11/validating-climate-models/>

<sup>14</sup> <http://www.cesm.ucar.edu/models/cesm1.0/cam/docs/port/pergro-test.html>