IS-ENES – WP3

D 3.6 – Remarks to a Unified HPC Environment

Abstract:

Grant Agreement number	**228203**	**Proposal Number:**	**FP7-INFRA-2008-1.1.2.21**
Project Acronym:	**IS-ENES**		
Project Coordinator:	**Dr Sylvie JOUSSAUME**		

Document Title:	Remarks for a Unified HPC Environment			**Deliverable:**	D3.6
Document Id N°:	D3.6	**Version:**	1.0	**Date:**	Oct 2012
Status:	Final				

Filename:	ISENES_D3.6_final

Project Classification:	Public, Confidential

Approval Status		
Document Manager	**Verification Authority**	**Project Approval**

REVISION TABLE

Version	Date	Comments	Authors, contributors, reviewers
0.1	June 2012	Collection of first ideas	K. Fieg
0.2	August 2012	2nd draft	G. Aloisio, J. Biercamp, K.Fieg, I. Epicoco, T. Jahns
0.5	September 2012	3rd draft	G.Aloisio, C.Basu J.Biercamp, K.Fieg, I.Epicoco, T. Jahns
0.9	October 2012	4th draft	Remarks by Reinhard Budich
1.0	October 2012	Makeover	Kerstin Fieg
1.01	November 2012	Minor corrections	J. Biercamp
1.02	February 2013	Review and minor addition	Marie-Alice Foujols

Table of Contents

From IS-ENES DoW

Task 3: Unified HPC Environment for ESM: DKRZ (4) (15 pm), CMCC (9) (7 pm), LIU (15) (3 pm), MPG (2) (M&D) (7 pm) Total: 32 pm

In this task we will improve the conceptual and technical integration of HPC centers to better serve the Earth System Modeling workflows. This extends from exchange of expertise and sharing of best practices over definition of common strategies and policies to actual technical integration of resources. In close cooperation with DEISA2 a specific unified environment for ESM will be provided that can be "loaded" when using a local, DEISA2 or remote high performance computer. This unified environment includes common modules, libraries, tools and policies (such as staging and transfer tools between different file systems, license management, distributed accounting management, co-allocation) as well as standardized path names, environment variables and the setup of a common username space. For the user, this “virtualization” will considerably facilitate the task of porting his or her applications to a new facility and of using several facilities in one workflow: He or she will not have to master the different working environments of all the different centers but will be able to use everywhere the same environment. In particular we will provide a road map of the efforts needed to efficiently port ESM workflows to PRACE using our unified environment.

As an intrinsic part of the unified HPC environment comfortable access to the distributed data archive developed in JRA4 will be provided. In a first step read access interfaces for data and metadata provided by SA2 will be integrated. Write access for data and metadata will be included stepwise by providing tool support to ease the appropriate metadata annotation and data formatting for data products generated by ESM workflows. Feedback on concrete experiences and requirements for archive access as part of the HPC environment will be provided to JRA4 and SA2.

Documentation and tutorials related to all components of the unified environment will be provided in an integrated and coherent way via the ENES portal

Summary

We discuss the growing demand for unification of the modeling environment and infrastructure to enable successful collaboration and intercomparison among the community. Therefore we want to give recommendation to HPC centers how to (optimally) support set up and run state of the art coupled ESMs on different platforms.

First steps have been made to standardize and unify data access and data description, which is a big step towards easier interaction and collaboration. Nevertheless there is a big gap in agreement on minimum standards for software development, definition of interfaces for model components, standard I/O and others.

This deliverable sums up the findings of the earlier deliverables D8.1, D8.2, D8.3, D4.1 and D4.2. Starting from here we draw conclusions regarding unification potential of the individual workflow steps.

Finally, a consequence from this deliverable with respect to available surveys, questionnaires and discussions is, that the structures and real life workflows are so heterogeneous among the community that a concept in the sense of having a “one fits all” box would not be easily accepted. Anyway, to make a step towards standardization users have to be educated about best practices and convinced by good examples of well-functioning, performing solutions.

1. INTRODUCTION

It can be expected, that the rapidly growing physical complexity and resolution of Earth System Models (ESM) will end up in a quickly growing need of computer time and data storage space. Furthermore, computing resources as well as services which are needed to drive the models and store the data are getting more complex.

In recent years we see a trend to answer the growing demand of resources with providing a gridded network of services all over Europe. As a result of this development, the normal scientist is often overstrained with the appropriate usage of the complex IT environment at large and the diversity of the various computer systems available in detail.

In IS-ENES, the requirements of the community in this context were collected and used as a starting point for an attempt to formulate a reference helping to foster the communication between researchers and computer sites; we call this document a “cookbook”.

One of the hardest problems for Earth System modelers is the difference in the environments found in different supercomputing sites around the world. Whereas there exist some – although not sufficient - standards in terms of operating, queuing, file and archive systems, there is no standard at all for Earth System models. So whenever a modeler or a model has to move to a new site, everything needs to be readjusted to the new environment.

In order to define a path towards a suitable HPC environment, interactions have been initiated in an earlier state of the project between IS-ENES members on the application side and DEISA and PrACE representatives on the HPC center side. Meanwhile DEISA does not exist anymore and the budget, previously foreseen to subcontract a DEISA partner was reallocated to CMCC contributing to the analysis of known and potential problems that arise when the Earth System Models are ported to PrACE sites.

The overall focus of WP3 / NA2 is to collect the requirements of the modeling community, compare it to the service, HPC centers can provide nowadays, discuss what can be done in the short and long-term and ease the communication between HPC centers and the Earth System modeling community. In particular the goal of Task 3 was to define a so-called “Unified Environment” for the commonly used Earth System Models among the IS-ENES community and then act as a kind of guideline for HPC centers to help them provide required resources and services as well as enabling the porting of applications.

In the beginning of IS-ENES it was assumed for Task3 of NA2 to establish a concrete technical realization as outcome of this networking activity. But results from D8.1, D8.2, D8.3, D4.1 and D4.2 showed it to be more useful for the community to gather the main results of the different surveys and discussion streams of the working groups, draw conclusions and translate them into operation guidelines for ESM developers and HPC centers.

Nevertheless, this document does not intend to comprehensively compile documents of models, couplers, workflows and tools again, but sum up the arguments of the various discussions and give distinct advice as to which tools are needed on an HPC platform to let an ESM run and ease its execution and handling.

While the aim of D8.4 (Report on Portability and Performance in IS-ENES ESMs) was to

improve the current performance of the ESMs on HPC systems and to understand what has to be done to improve performance on future HPC systems, we want to sum up the requirements of the climate community and deliver a kind of cookbook to help the HPC centers to create the environment needed to run an ESM and produce results.

For that, the first part extracts the relevant results from previous deliverables and breaks down into subtasks the complex ESM community workflow discussed in NA2 Task 2 into subtasks. In the second part, the cookbook, we start with matching the prototype workflow from part 1 with real life procedures of the climate community. Finally this should lead to a recommendation from which we think they can be reformulated in a reusable way as a first step to a unified environment.

2. COLLECTION OF QUESTIONIARE RESULTS

The reality is (from ENES – foresight, S. 15):

An Earth system model is usually driven via a complex infrastructure of scripts which acquire parameterization and boundary data from storage, extract or perhaps compile some specific code, and then bind that code to the data and specific parameters (usually via "namelists") before executing it. The final model configuration will normally be specifically targeted at one type of parameterization (for example, using a specific decomposition of the model grid onto a specific number of processors) for a specific piece of hardware, and thus cannot be easily reused. Typically the software infrastructure to do this is tightly coupled to the model software itself ("custom designed"), with the tools for parallelization embedded within the scientific code.[1]

2.1 MODELS & APPLICATIONS

This collection sums up the main results from Deliverables D4.1 and D8.1.

D4.1 compiled the existing documentations for European ESMs. Meanwhile, the individual documentation has been updated to a more formalized and standardized description following the Common Information Model (CIM) developed in the METAFOR EU project. The detailed documentations can be found at the ENES – portal, which harvests the information from the METAFOR portal (https://verc.enes.org/models/earthsystem-models)

In D8.1 the so called IS-ENES evaluation suite is defined and described. It consists of the following models

- CMCC-MED from CMCC (I)
- ARPEGE-NEMO from CERFACS (F)
- IPSLCM5 from CNRS-IPSL (F)
- HadGEM from MetOffice (UK)
- MPI-ESM from MPI-M (D)

One of the main results of D8.4 is that having all the required libraries and applications already implemented by the default on the new target platforms significantly eases the porting and setup of models in general.

The following collection assembles the questionnaire results from deliverable D8.1 regarding programming language, tools and libraries required to run the models of the IS-ENES Evaluation Suite.

1. Program Languages

- C

[1] https://is.enes.org/the-project/communication/ENES%20foresight.pdf/view

- Fortran 90/95/2003

2. Libraries
- NetCDF
- MPI 1 / 2 (MPIch, OpenMPI)
- OpenMP
- BLAS
- Lapack
- xml
- GCOM (needed for HadGEM)
- Mass (needed for HadGEM)

3. Divers Tools
- svn / cvn
- git

2.2 WORKFLOWS

Summing up the collection of answers to the questionnaire, which is part of deliverable 3.2 and 3.5, leads to a reference workflow, which maps the usual steps from data production to long term archiving. Starting from that, we break down workflow and rearrange them into subparts, which are general and comprehensive enough to meet the needs of a large range of ESM activities. The detailed result of the survey can be found at https://verc.enes.org/computing/workflows/.

In more detail, the part of the generalized workflow we focus on consists of

- Model parameterization
- Defining and providing initial and restart files
- Defining necessary processing steps
- Retrieval of model source
- Defining model setup (include / exclude physical processes)
- Configuration of the process chain
- Generation of scripts and job environment
- Compiling
- Linking
- Processing executable
- Model launch
- Providing of computer resources
- Post processing
- Quality check
- Creation of metadata

- Data archiving

Some parts of the workflow cannot be standardized or operationally supported by e.g. HPC centers, because they will always require research or at least manual intervention done by the modeler. Apart of the tasks whenever scientific questions are touched, a close interaction between modelers and HPC centers / computer engineer is needed to complete the described workflow.

. While the model development and parameterization parts of the workflow as well as the scientific quality checks cannot be unified, standardization is desirable for the more technical tasks to make model porting easier and general purpose computing centers more attractive to the HPC community. Besides the technical configuration of the experiment and launching the model execution, the HPC centers can support the workflow by delivering the parallel environment, caring for the load balancing of the applications, installing the basic tools for data evaluation and standard post processing and finally taking care for short, medium and long term data storage

2.3 COUPLER-SOFTWARE

A task holding high potential for unification is the coupler. It generally enables the interaction of model components to finally end up as “Earth System model” (ESM). When the whole climate research community would use the same standard coupler, including, excluding and exchanging components would be possible in a defined and unified way. This was one of the main ideas of the PRISM project (Program for Integrated Earth System Modeling), founded by the European Union (Dec 2001 - Nov 2004), where 17 European climate research centers and 4 members of the computer industries started to develop an infrastructure for a European climate research network. One outcome of PRISM is the community coupler OASIS3 used not only by the European climate community.

Nevertheless, the following list is a very incomplete collection of tools which are currently used for coupling in the community – every of them serve special requirements. The most commonly used ones are:

- ESMF: Earth System modeling Framework (http://www.earthsystemmodeliing.org) (USA)
- CPL7: NCAR, Boulder (USA)
- FMS: The Flexible Modeling System, GFDL, Princeton, (USA)
- MCT: Model Coupling Toolkit
- OASIS3 / OASIS4 / OASIS3-MCT (EU)
- BFG: The Bespoke Framework Generator (http://www.cs.manchester.ac.uk/cnc/projects/bfg) (GB)

For more details see the overview paper Valcke et al.[2012].

It is obvious that installing and maintaining all the coupling tools is not reasonable for HPC centers. To lower the hurdle for changing the platform will be significantly eased by an agreement on one (or a small number) of tools.

2.4 ENVIRONMENT

To discuss the unification potential of the environment the starting point are the deliverables D8.1 – a collection of models and their requirements regarding libraries and architecture - and deliverable D8.4 – the description of experiences with porting the ESMs mentioned in D8.1 to different HPC systems.

The most important conclusion of the porting tests of D 8.4 had been that neither the switch of one ESM between different platforms nor driving different models at one platform "had been too problematic technically in general". Nevertheless, there had been significant technical problems in the details. We can subdivide the problems into four parts:

Part 1: One ESM consisting of a number of climate components

Every ESM model component requires an individual collection of compiler flags, strongly dependent on the target platform. This makes porting always complex and will require customized tuning.

Nevertheless up to now, it seems that all climate components of one coupled ESM use at least the same version of compiler, libraries and tools.

➔ Unification can be done, but would require: a wrapper script for every component containing all compiler flags for every target system and all the commonly used OS versions.

Part 2: Different ESMs on one platform

A core problem is that the different ESM codes have completely different file structures. As a consequence

- The overall directory structure of different ESMs is far away from being similar
- Code pieces or climate components (e.g. OASIS, NEMO) which are part of various ESMs, are not located at the same place in the directory structure.
- Code pieces or climate components (e.g. OASIS, NEMO) which are part of various ESMs are adapted for optimal use in the specific ESM during the time: the code differs.

➔ Unification can be done, but it is rather unlikely that the effort to re-standardize a tool like OASIS (which once was designed as a standard tool but then was changed according to local requirements) and store it usable for all models in a central place, will be honored in the long run.

Part 3: Different platforms

It was mentioned in the survey, that at some sites necessary libraries and applications were not implemented by default (e.g. NetCDF, Blas, Lapack ..), so that porting can be hampered due to "wild" installation of software

➔ Unification can be eased by prescribing a minimal software stack required to run models from the IS-ENES evaluation suite

Part 4: One ESM on different platforms

Resulting from above findings: the same ESM can require different compiler flags for optimal use

- Dependent on the platform selected (from different vendors e.g. IBM, Cray, NEC, Oracle….)
- Dependent on the operating system (aix, unicos, linux…) and the release version
- Dependent on hardware configuration (available number of CPUs / Nodes, processor type / Interconnect..)
- Dependent on the components or version of components (e.g. OASIS3 vs. OASIS3-MCT) for coupling

➔ Unification can be done by defining unified configuration files with the required information of all climate components, all software and all platforms. This will require a lot of manual intervention and has to be maintained regular to be up to date with changing hardware and software development in the community and at the HPC sites.

It has to be emphasized, that all efforts towards a unified environment does not avoid to finally ending up with executables, which deliver non bit-identical results on different platforms. Of cause, this is even truer for ESMs running on different platforms in non-unified environments.

3. THE COOKBOOK

3.1 GENERAL REMARKS

The very final goal of all the effort combining model code and complex IT resources must be to deliver scientifically meaningful results. The technical bottlenecks for the collaboration between scientists and resource provider have been discussed above.

Here we intend to deliver a kind of guide how to set up a state-of-the-art ESM and such describe the potentials for unification.

We see two different types of applications which in future will absorb the available resources:

- Workflow type 1(“Capability”): large scale ESMs with very high resolution in space and time and increasingly complex physics
- Workflow type 2 (“Capacity”): a medium complex ESM running in medium to high resolution in parallel to create a large number of ensemble runs. The ensemble runs can be set up in a way, that communication processes between the ensemble members are required, e.g. due to collective data processing or synchronizing data during the runtime.

The different types of applications will require different strategies to solve their problems and run efficiently.

For both types of workflows a technical unification – even of parts - is only possible within certain limitations. We have to accept that even a minimal agreement regarding tools, modules, workflows will always influence the performance dependent on the target platform / vendor (NEC, IBM, Cray ..) / OS (linux, aix, UNICOS, …) as well as on the model resolution of the setup and physics included in the model.

In recent years national and international projects were initiated (e.g. METAFOR, C3 Grid, UNICORE …) to make first steps into the direction of a unified environment. To save manpower and benefit from their work, the tools and software developed there should be reused as far as possible in close cooperation with the developers.

We start our discussion with focusing on best practices at exemplary sites – here: DKRZ and CMCC. The workflows described here give a good impression, in which respect setups differ and how deep the processes are usually wrapped into locally grown scripts, habits and tools. It is obvious, that replacing at least parts of the traditional process by introducing unified sub tasks will necessarily end up in a comprehensive revision of the total workflow.

3.2 CAPABILITY RUNS VS. CAPACITY RUNS

The usual way to speed up code performance is identifying and optimizing those parts consuming most of the computing time. According to Amdahl’s law, the speed up of the code is limited by the time needed for serial parts of the code. For the ESMs from the IS-ENES evaluation suite, except HadGEM (that is MPI-ESM, CMCC-MED, ARPEGE-NEMO and IPSLCM5), one of the bottlenecks is the serial coupling tool OASIS3, which will

hamper a massive parallelization. This hard limitation should be kept in mind thinking about porting real life climate codes in very high resolution to massive parallel computer environments like PrACE and running them in a Capability workflow. As long as this serial part is not replaced, we expect only limited benefit from this effort.

On the other hand, HPC partnerships like PrACE can be of great advantage when workflows of the Capacity type are set up. As highlighted above, the limited potential of parallelization will not be of consequence for the performance when running a large number of similar experiments in parallel on distributed computer resources. This should be a step forward in taking advantage of their possibilities. Furthermore, for Capacity workflows a unification of commonly used code parts is less important than unification of hardware, modules and tools. To be in line with PrACE requirements, processing of shared intermediate data, can be placed inside a wrapper MPI code to benefit from the fast network between nodes.

It is a common requirement, that bit identical results should be reproduced. This should not be a problem to guarantee, when running a large number of ensembles on one uniform target system – at least as long as compiler versions and hardware do not change. The case is different when launching ensembles on different architectures (e.g. in a compute grid) and comparing the results. Due to the heterogeneity of the different hardware the bit-2-bit equivalence among simulations cannot be guaranteed even if the same software environment (compiler version, OS, libraries) is used. We think, it has to be discussed communitywide in future, up to which degree an eventual discrepancy among results from different architectures for Capacity workflow runs should be tolerated, because ensemble experiments are designed to be combined and analyzed statistically anyway. Nevertheless, influence of the hardware on the results has always to be clearly very small compared to the signal to be interpreted.

3.3 GENERAL WORKFLOW

During the last years a number of tools supporting scientists and computer programmers in their daily work were developed. On the one hand there are initiatives like the *Kepler* project, who developed a scientific workflow system (https://kepler-project.org/) that supports the creation, execution and sharing of models, software and results, and on the other hand there are tools like *cylc*, a process meta scheduler (http://hjoliver.github.com/cylc/), that makes it easier to set up, control and maintain modeling environments and workflows. Such tools can be a step into unification of the environment, when a number of institutions agree on the support and the use. Nevertheless actually, we see various custom-made approaches among the community.

Cross referencing to the reference workflow described in chapter 2.2 we will follow two of those home grown approaches and describe, how the tasks are treated here. Therefore, we group the general workflow from chapter 2.2 into three blocks:

Blocks discussed here	**Matching tasks from chapter 2.2**
defining and preparing the environment	• retrieval of model source • defining processing steps • Configuration of processing chain

	• Generation of scripts and job environment • Compiling and linking the code • providing computer resources
model initialisation	• model parameterization • defining model setup • defining and providing initial and restart files
model run with post-processing and archiving	• Processing executable • model launch • Postprocessing • Quality check of data • Creation of metadata • Storing, archiving and publication of data

The following discussion does not distinguish between the workflows of the Capability and Capacity type.

A. DEFINING AND PREPARING THE ENVIRONMENT

All steps have to be done in the forehand of a model run. The block “Defining and preparing the environment” includes both technical and conceptional work steps.

- The technical work steps can be done standardized and should not influence the scientific statement. In particular this can be “retrieval of model source”, “generation of scripts and environment”, “providing compute resources” and “compiling and linking”.
- On the other hand, there are the conceptional work steps that may have influence to the scientific proposition. Among those can be “defining the process steps” and “configuration of the process chain”.

The latter steps need the interaction of technical together with scientific staff to ensure a meaningful result (see later).

I) DKRZ, general remarks

To set up and run coupled ESMs at DKRZ a home grown product, the “integrated model and data infrastructure” IMDI is widely used.. IMDI is an overarching tool, which generates the complete environment as well as all processing scripts for compiling, running,

CMORising, archiving and post processing for the complete MPI-ESM model run. IMDI has to be retrieved from a software repository (SVN / git) together with the latest revision of the model code. This guarantees the creation of the required general directory structure and secures the retrieval of a defined revision of the model code. Furthermore, IMDI secures the traceability of the experiment: if the scripts are stored after using, the collected information can easily be transferred into a set of provenance data. Model code and IMDI combined constitutes the necessary structure to run an MPI-ESM experiment.

IMDI is designed to be used on different (commonly used) platforms but requires careful updating of the software to stay up to date with compiler / library / OS updates. Nevertheless, IMDI is relatively easy adaptable to other platforms – because only technical work steps are necessary - but would require a lot of rework to make it usable for other models than MPI-ESM – because here conceptional and individual reword has to be done for the every ESM. Currently DKRZ and MPIM, based on the experience made with IMDI, are evaluating new approaches based on tools like “cylc” (http://cylc.github.com/cylc/) for their potential to provide a more widespread and more lightweight solution to reduce maintenance effort.

II) In Detail

1. Retrieval of the model sources

CMCC & DKRZ

An authorized user retrieves the finalized model code, prepared in the step above, from the software repository.

2. Configuration of the process chain

CMCC

At CMCC the user can run the model on the platforms on which it has been configured in the step before or select another one. In this last case, a configuration phase for compiling and running the code is necessary. A tool is under development at CMCC to discover the available climate models through the definition of basic metadata: model name, version, keywords etc.

The user can choose among different experiments such as ESM, ensemble, etc. The job execution chain requires the writing of a run script which specifies: the structure of the execution, inputs outputs logs and restart directories; the simulation period and the chunks duration with the restart mechanism; the post-processing; the archiving. The configuration step consists in the run script definition. Currently, each modeler defines the process-chain without a common template and using textual editors. A graphical user interface is under development at CMCC to support the composition of the process chain acting on a predefined workflow skeleton.

DKRZ

At DKRZ the user can easily configure and run the model on platforms, which are incorporated in IMDI. If other target platforms should be used, an adaption of IMDI is necessary with respect to local structures and conditions: IMDI structure allows the adjustment to specific requirements like target machine (IBM vs. Linux cluster – aix vs

linux), the specific model set up (e.g. coupled vs. stand-alone), experiment characteristic (e.g. restart vs. spin up), including or excluding post processing or different levels of quality check etc.

The length of the individual job chunks is dependent on the configuration of the queuing system and actual status (memory, time, number of CPUs available). For optimal use of the environment, the total experiment is divided into smaller time spans called in a loop.

The adjustments are set in a configuration file, which than generates all scripts for running, post processing, archiving the experiment (e.g. experiment.run, experiment.post, experiment.arch).

3. Compiling, linking and processing executable

CMCC

During this step the modeler should already have the right configuration for the compiler and linker. He has only to select the right architecture among those already supported by the model. If the model has been never compiled on the target architecture, eventually the existing run jobs and source codes have to be adapted and checked in into the repository with the new configuration for the target architecture.

DKRZ

The IMDI provides a script for compiling and linking a defined reversion of the model source code and takes care of the platform dependent settings (compiler, linker…). Finally an executable for every model component is processed, so here we finally end up with a number of executables, e.g. echam.x, mpiom.x and oasis.x.

4. Providing computing resources

CMCC

The computational resource requirements depend on different factors: type of experiment; configuration of the model; load balancing of different components. The resource requirements are defined within the run script before the execution using the directives for the local scheduler. The run script, defined in the parameterization step, has to be modified in two cases: the computing resource requirements must be defined for the production run; the target architecture is not in the list of the supported architecture in this case the run script will be improved adding the scheduler directives and submit commands for the new architecture. In this latter case the source code repository of the model is updated with the configuration of the new architecture.

DKRZ

The hardware requirements needed to run MPI-ESM are dependent of the specific model configuration and resolution. As example: to run the state-of-the-art coupled model in a low resolution configuration MPI-ESM-LR (ECHAM6.1: T63L47 / MPIOM: GR15L40) one needs:

- Processor: approx. 180 CPU*h per year model run in a IBM power6

- Storage : 33 GB output per year
- Memory : 700 mb

B. MODEL INITIALISATION

The block “model initialization” hides a lot of complex processes which have to be considered itself as sub workflows. On the one hand there are tasks like the parameter tuning, which can be seen as an iterative process during which the modeler refines and tunes the model physics internally, estimating the parameters and often also modifying the code. On the other hand there are processes like defining a model setup which basically means to secure, that the needed processes are included in the model run and treated in the correct way. The validation of the new model version requires several short test runs followed by basic post-processing for a preliminary analysis of the results**.** Furthermore initial and restart files have to be defined and provided accorded to the experimental set up.

The listed tasks are highly individual dependent on the site, model and project that they cannot be easily unified.

CMCC

1. Physics parameterization: in this step, the modeler acts on the parameterization of some physic phenomena. If these phenomena have been already modeled, the modeler only performs the parameters setup in the configuration files. Otherwise, he modifies the code to model the phenomena not already considered. This step requires the expertise of a climate modeler and often represents the heart of the scientific production of the modeler. There is not any specific tool used for modifying the source code and the configuration files. A text editor is commonly used.

2. *Configuration of the process chain for validating the parameterization:* during this step the complete script for running the model and getting basic post-processed results is defined and written. The process chain aims only at validating the model parameterization. The definition of the process chain is a responsibility of the climate modeler.

3*. Compiling, linking*: the model has to be compiled on the target platform to be executed. During this step the modeler has to define the best compiler options to get an optimized binary for the target architecture. Some models (nemo, oasis, etc) provide tool for easily define the compiler and linker options for different platform and to easily switch from a platform to another. Often these models include also a set of common compiler options for most used platforms. The definition of the best and tuned options for the compiler is currently often managed directly by the climate modeler but it should be done by the system administrator. The procedure for compiling and linking is quite standard and allows generating the executable for running the model.

4. *Providing computing resource and execution*: a run script to launch the model has to be written. It includes the directives for the LRMS (Local Resource Management System) installed on the target machine and the resources required for executing the job in terms of memory and processors/cores. The definition of the run script requires a deep knowledge of the target platform and how the scheduler handles the resources (e.g. binding of

processes on the physical cores, memory affinity..). The optimal values for these last parameters can be retrieved due to a preliminary scalability analysis on the target machine. For the coupled model, an algorithm for the resources load balance among the model components is used in order to minimizing the total execution time.

5. *Basic post-processing for validation*: the outputs of the model execution are post-processed to become human-readable. Produced data allows the modeler to validate his parameterization choices (at step 2).

6. *Repository update:* the validated model is uploaded into the git repository and published for the CMCC community, ready for production runs. A new graphical interface is under development (internally at CMCC) for publishing and retrieving the list of all of the climate models and the pre- and post-processing tools internally available (through git repository).

DKRZ

Using the IMDI environment and its prescribed procedures, the source code relevant for the explicit experiment setup has to be tuned, tested and provided. This has to be done individually for every experiment (or at least experiment group / ensemble) by the model developer / researcher.

The steps include the configuration of

- Model parameterization,
- Validation of physical processes
- Securing correct parameter setting
- Defining and providing initial and boundary conditions

The final code revision has to be checked in into a model repository and checked out on the target production machine.

Because this task is an essential part of research, we don't see potential for technical unification - except ensuring and providing a stable and defined environment.

C. MODEL RUN WITH POST PROCESSING AND ARCHIVING

Like block A, the block "Model run with post-processing and archiving" consists of technical and conceptual work steps. The technical work steps span from "processing executable", "launching model", "Creation of Metadata" to "storing, archiving and publication of data". The more conceptual work steps are "postprocessing" and "quality check of data".

1. Processing batch job and experiment launch

CMCC

Currently the execution is started by submitting the previously defined run script using remote shell terminal. The modeler monitors the execution inspecting log files and of the partial results. At CMCC for these activities a graphical interface is also available.

DKRZ

After creating all jobs scripts for monitoring (if required), post processing, CMORising and archiving etc, all elements of the processing chain are available and can be combined. They will be launched by the main batch job script called *experiment.run*. It drives the complete experiment on the target machine from execution to archiving. At DKRZ the target machine is usually an IBM power6 supercomputer.

2. Post processing

CMCC

One or more post-processing activities can be defined depending on the model in question and the processing output format. For each one, an operations chain is defined within a post-processing script, which has to be equipped with LRMS directives and executed on the target machine.

DKRZ

After finishing one of the modeling time slices, a post processing job is launched followed by the CMORising job and an archiving job. This second processing part can run in parallel to the modeling part. The processing steps are called here are coordinated by the overarching execution script.

3. Data archiving

DKRZ & CMCC

Output data are archived following a standard procedure defined on the management policies of the CMCC and DKRZ supercomputing center, regarding requirements concerning e.g.

- File names,
- Directory structure
- Essential metadata
- Data formats
- File sizes

At DKRZ, the archiving process is coordinated by the overarching execution script, mentioned above.

4.3 IDENTIFYING REUSABLE SUB TASKS

The workflows described above are real life examples and show the deeply inter-woven interaction of infrastructure and code structure. To replace at least sub parts of the process will very likely end up in a comprehensive revision of the existing workflow.

Cross referencing the description above, we aim to identify sub tasks, which theoretically can be formulated in portable entities to be easy reusable on different machines at different sites.

Again, we will separate between the tasks that we identified as technical work steps and conceptional tasks.

Block A: Retrieval of the environmental infrastructure and software

1) technical

i. **Retrieval of model source:** The procedure of retrieval itself can be standardized and unified by using revision control tools like portable and open source products like svn / git /cvs. This is advised anyway because it enables tracing the software development and secures the use of defined releases, tags or reversions.

ii. **Compiling, linking and processing executable**: A discussion is needed to find a general agreement about the requirements necessary for the individual models to ease the porting to different architectures. The final goal should be that optimal compiling options for the specific target platform have to be defined and – if necessary - tuned by the computer engineer or system administrator.

iii. **Providing computing and data resources**: A good example for unified environment and the consequences for software development is the trend to provide computing and data resources in a gridded environment. Distributed systems are connected and reachable for the user by a single sign on. To benefit from general development, the used software – and this means model code and workflow environment - has to be as easy portable and flexible as well as robust and fail tolerance as possible.

2) Conceptual

i. **Configuration of process chain**: Defining common sub tasks should be possible, but will require comprehensive revision of the individual workflows at every specific site. In order to “standardize” the configuration of the process chain a flexible tool should be provided. The tool - implemented as a shell script or with a graphical user interface - should implement the general workflow providing a skeleton or template easily configurable by the modeler for his specific experiment.

3) Summary

The general potential for unification of the infrastructure environment and software is high but difficult to push. More standardization across the community can only be achieved, with general agreement what best practices in the field of infrastructure design finally are. The easiest way of winning users is by providing good examples and educates them using the available tools. Nowadays, every site or even every researcher uses its own - historically grown – environment and has to be convinced to use a new to designed unified environment by providing well-functioning and practical solutions. It has to be obvious, that the benefit has to be bigger than the effort to reorganize the established workflows.
Later we discuss the IPCC example, where this way to finally reaching an operational standard worked out well.

Block B: Model initialization

1) Summary

The procedure of model initialization can only be formalized and standardized in a limited way, because it is mainly related to scientific work which therefore is per definition individual. Nevertheless, the formal process for physical parameterization, model setup or deriving initial and boundary files could be supported by unified tools like version control systems. This could document the process, ease metadata production and make the final result more reproducible and usable.

Block C: Model run with post-processing and archiving

1) Technical

i. **Processing batch job and experiment launch**: The experiment launch strictly depends on the target architecture and on the local scheduler. A set of files for different platforms with the directives to the local scheduler have to be provided, to enable an easy switch of computer architectures. Nevertheless, at least the definition of unifiable sub tasks should be possible, but will require general agreement, willingness to use and finally comprehensive revision of the individual workflow.

ii. **Data archiving**: usually the archiving procedure of large data sets follow the standard procedure defined on the management policies of the data centers e.g. regarding data format, metadata and sizes. Increasingly, big data centers agreed on mandatory standards depending data and metadata. One goal was to enable a worldwide defined data distribution a secure life cycle management of the data. For that, world data centers were founded like WDCC (DKRZ), WDC-GMG (NOAA), WDC-MARE (AWI, Germany).

2) Conceptual

i. **Post processing, quality checking of data:** as mentioned above, in theory here is unification potential, but in practical it would require a communitywide agreement on standards and tools. In former times this effort looked hopeless, because of individual requirements, technical boundary conditions, and personal preferences. But today, we have an example, where enforcing a standard worked out. We discuss it below.

3) Summary

As mentioned above, there is a successful example of standardization and unification that has to be highlighted here. It was a general requirement to all institutes which intent to participate in the IPCC AR5 report to obey on an agreed and defined standard for the resulting products in respect of variable, data formats, timestamps etc. Finally, this should guarantee the uniform processing of data and better comparability of results.

The rules affected the workflow in data post processing, quality checking, CMORising and structures of archiving. In the beginning, there was not enthusiasm everywhere, because massive rework of the grown procedures was enforced, but at the end, the benefit for the

scientists was measurable and more communities are convinced to follow the example. Consequently, now there is a trend to stick to the IPCC procedures for follow up projects.

4.4 UNIFIED ACCESS TO METADATA & DATA

There is a growing need for data to be shared and compared. Referring to the above, providing standardized interfaces for data and metadata retrieval makes life easier for the scientist. This is not possible without agreed-upon metadata and data standards, regarding wording and content but also quality. Furthermore, well defined and standardized data access - at least for reading - is necessary.

Unified data access is e.g. the focus of the German projects C3Grid and C3-INAD, federating climate data from German research institutes and providing them in a gridded environment. The access is granted via single-sign on to the trusted user from the community.

For metadata, the EU project METAFOR made a step towards unification. The goal was to ease the production of unified metadata by formulating a Common Information Model (CIM) for climate data and the models. The work consisted of two elements: the CIM, which describes the data and the models in a standardized way and a questionnaire, which ensures and eases the input of the required information by the user.

Another example for both, a successful unification of data access and metadata description is the IPCC strategy, mentioned above. Here it is obvious, that defined interfaces to exchange and defined metadata are the precondition for international collaboration and comprehensive analysis.

Having these examples in mind, we see strong progress in the field of unification of data and metadata and suggest continuing the efforts.

5. CONCLUSION AND DISCUSSION

There is an unquestioned demand to unify the modeling environment and infrastructure to enable successful collaboration and intercomparison. Models or code parts have to be shared as well as data and results. Therefore defined interfaces, metadata and quality checked data are essential. This is only possible if the scientific community agrees upon in certain standards and is willing to accept changes in their traditional workflows. There are examples - like the IPCC protocol – where the results are promising and at least parts of the community intent to stick to them for future projects. It seems obvious that unification and standardization are only useful and beneficial if a large fraction of users accept them and use the solutions based upon them.

Where are potentials for standardization?
Generally as discussed before, unification potential can be found at every step of the workflows.

What is done?
First steps have been made to standardize and unify data access and data description, which is a big step towards easier interaction and collaboration.

What can be done?
We see a big gap in defining minimum standards in software development, e.g. defining interfaces for model components, standard I/O and others. Moreover the workflows from retrieving code and environment to archiving the results are far away from uniform. Nevertheless, the structures as we see them now have historical reasons on the one hand, and are resulting from restrictions of the target platforms.

What is realistic?
One of the most important conclusions of the surveys, questionnaires and discussions in the community regarding unified environment is that hardware, software and environmental details of the models, research tasks and HPC sites are so diverse, that the potential to define and create a uniform environment is limited. Furthermore, a concept in the sense of having a “one fits all” box would not be accepted in the community.

We think, discussing the real life DKRZ workflow for MPI-ESM gives a good example, how deeply the processes are wrapped into locally grown scripts and tools. To replace at least parts of the traditional process by introducing unified sub tasks will necessary end up in a comprehensive revision of the workflow.

In our opinion one of the most important conclusions that can be drawn is, that an environment that is provided by HPC centers (directory structure, libraries, tolls, compiler versions…) has to be as stable as possible during a long as possible time span, at least for the lifespan of a project (which typically is in the order of many months to a couple of years!).

The reason is that the overall time needed for

- Porting a code

- Physically tuning the code
- Optimizing the performance for a special architecture / machine
- Optimizing the code for a special physical set up

is so time consuming and special for every target platform, that it only makes sense if one can rely on this customer tailored configuration for a sufficiently long time.

6. ACRONYMS & GLOSSARY

Appendix A : Acronym	Definition
ESM	Earth System Model
DEISA	Distributed European Infrastructure for Supercomputing Applications http://www.deisa.eu/
PRACE	Partnership for Advanced Computing in Europe http://prace-project.eu/
HPC	High Performance Computing
NEMO	Nucleus for European Modeling of the Ocean http://www.nemo-ocean.eu/
MPI-ESM	Max – Planck – Institute Earth System Model (http://www.mpimet.mpg.de/en/science/models.html
HadGEM	
OASIS	Ocean Atmosphere Sea Ice and Soil coupler https://verc.enes.org/models/software-tools/oasis/
CIM	Common Information Model, developed by FP7 project METAFOR
UNICORE	Uniform Interface to Computing Resources (http://www.unicore.eu)
SVN	Subversion, open source revision software http://subversion.tigris.org/
Git	Open source revision software (http://git-scm.com/)
v.E.R.C.	Virtual Earth System Modeling Resource Center
METAFOR	Common Metadata for Climate Digital Repositories http://metaforclimate.eu/
cylc	Cylc is a suite engine and a meta scheduler http://cylc.github.com/cylc/

Table 1: Glossary of Acronyms