Geosci. Model Dev. Discuss., 7, 4429–4461, 2014
www.geosci-model-dev-discuss.net/7/4429/2014/
doi:10.5194/gmdd-7-4429-2014


Geoscientific Model Development Discussions
Open Access



# Enhancing reproducibility of numerical simulation result on the C-Coupler platform

**L. Liu[1], R. Li[2], C. Zhang[2], G. Yang[2,1], and B. Wang[1,3]**

[1]Ministry of Education Key Laboratory for Earth System Modelling, Center for Earth System Science (CESS), Tsinghua University, Beijing, China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]State Key Laboratory of Numerical Modelling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China

Received: 24 June 2014 – Accepted: 9 July 2014 – Published: 17 July 2014

Correspondence to: L. Liu (liuli-cess@tsinghua.edu.cn), G. Yang (ygw@tsinghua.edu.cn), and B. Wang (wab@tsinghua.edu.cn)

Published by Copernicus Publications on behalf of the European Geosciences Union.







## Abstract

Reliable numerical simulation plays a critical role in climate change study. The reliability includes the technical reproducibility, i.e. bit-identical results of numerical simulation can be reproduced. It is very important for model development and scientific researches but has not been satisfactorily addressed yet so far. To address the technical reproducibility, necessary information about it is firstly analyzed, and how to enhance it on the Community Coupler (C-Coupler) platform, a uniform runtime environment that can operate various kinds of model simulations in the same manner, is then detailed. Moreover, we share a series of experiences and suggestions with scientists and model groups for achieving the technical reproducibility. We believe that, the proposed implementations, experiences and suggestions can be easily extended to other model platforms, and can prospectively advance model development and scientific researches in future.

## 1 Introduction

Numerical simulation plays a critical role in understanding the past, present, and future of climate. With the rapid advancement of science and technology, a number of numerical models have been developed for climate simulations, predictions and projections, including atmospheric models (Dennis et al., 2012; Li et al., 2013b; Skamarock et al., 2005), land surface models (Ji and Dai, 2008; Oleson et al., 2004), ocean models (Griffies et al., 2010; Liu et al., 2012; Smith et al., 2002), sea ice models (Hunke et al., 2008; Lipscomb et al., 2009), wave models (Booij et al., 1999; Yang et al., 2005), and coupled models such as climate system models (CSMs) and earth system models (ESMs) (Collins et al., 2006; Gent et al., 2011; Giorgetta et al., 2013; Hurrell et al., 2013; Li et al., 2013a).

To help the model development and scientific researches on climate change study, an increasing number of international model intercomparison projects (MIPs) have





GMDD
7, 4429–4461, 2014



been proposed, which attract more and more model versions to participate in. Each MIP usually provides a set of standard simulation experiments for running, evaluating, understanding and improving the models. The simulation results from the MIPs are always open to the whole world for climate studies. One typical example is the Coupled Model Intercomparison Project (CMIP). In the CMIP Phase 3 (CMIP3), less than 30 coupled model versions participated, while in the CMIP5, there are more than 50 coupled model versions. Now the simulation results from the CMIP5 can be freely downloaded and have been widely used for various scientific researches.

One important target of model development is to make the models able to reproduce the past climatic phenomenons, so as to convince their capability in predicting / projecting future climate. This kind of reproducibility can be classified as scientific reproducibility. A lot of works have been conducted to improve and evaluate the scientific reproducibility of models. As a twin brother of the scientific reproducibility, the technical reproducibility, i.e. bit-identical results of numerical simulation can be reproduced, is comparably important but has been paid much less attention. To achieve the technical reproducibility of a simulation, a lot of related information needs to be recorded. This is a very tough work. For example, information about the experimental settings given in most of scientific papers is not detailed enough for reproducing the corresponding simulations by other scientists. Even though the experimental settings are detailed enough, bit-identical results can hardly be reproduced and a large amount of works are required for the reproducing. Regarding the MIPs, none of them emphasizes the technical reproducibility of the simulation corresponding to the submitted results. This study will focus on the technical reproducibility. In the following context, the word *reproduce* refers in particular to the technical reproducibility.

To the best of our knowledge, there are almost no existing works addressing the technical reproducibility for earth system modelling. One possible reason for this situation is that the technical reproducibility does not directly target the scientific reproducibility. However, we believe it is very important for model development and scientific researches, due to the following reasons:





GMDD
7, 4429–4461, 2014



1. Improve the efficiency of model development. To develop a model and improve its scientific reproducibility continually, scientists always need to constantly repeat the same kind of simulations, with different versions of model code, different initial conditions and forcing data, different parameterization schemes, different parameters, etc. Due to the lack of storage space, the results from most of simulations could not be kept permanently. Along with the rapid development of science and technology, it is possible that some insignificant simulation result that has been removed from the storage may become valuable and significant in future. In other words, it is unavoidable that scientists want to reproduce some previous simulations. Moreover, it is possible that scientists forget which simulation produces a given data file among a big dataset, and this simulation result may hardly be reproduced. Therefore, through enhancing the technical reproducibility, the efficiency of model development can hopefully be improved.

2. Promote the collaboration on model development. There are an increasing amount of scientific and technical challenges to be addressed for model development, especially for developing CSMs and ESMs. Consequently, model development more and more depends on the collaboration between different groups and different institutions. Given that several model groups cooperate together to develop an ESM and each group is responsible for a component model, it is unavoidable that one group wants to reproduce some model simulations done by other groups, due to the interrelations among component models in a coupled model. Therefore, the collaboration between different model groups can be further promoted through enhancing the technical reproducibility.

3. Improve knowledge sharing and spreading. Model simulations can be regarded as a book of knowledge. An effective approach for training new model members is to ask them to reproduce a set of existing simulations, study old conclusions and even draw out new analyses. Moreover, scientists may want to reproduce some model simulations from scientific papers, and then improve the model and





GMDD
7, 4429–4461, 2014



experimental settings for further researches. Through enhancing the technical reproducibility, the knowledge intra the simulations can be shared and spread more easily and efficiently.

4. Improve the reliability of models. Numerical models are a class of scientific programs. With the rapid development of models, the number of code lines of models gets larger and larger, especially for CSMs and ESMs. It is unavoidable that there are scientific and technical bugs in the model code. Testing is an effective approach for detecting and fixing bugs. Generally, if there are more test cases, more bugs can be detected and fixed, and the programs will become more reliable. MIPs play an important role in testing models through model comparisons, which have already provided a lot of test cases for model groups. The scientists and engineers who are not in model groups can also contribute to model testing. If the simulation results published on scientific papers can be easily reproduced, much more test cases will be conducted for model development and models will become more reliable for scientific researches. Moreover, the conclusions in scientific papers will also be more trustable.

It is a challenging work to achieve the technical reproducibility for various kinds of models. The most important reason is that the bit-wise model simulation result is very sensitive to a lot of factors, including computers, number of processors used, calling order of subroutines of physical processes, compiling options, external forcing, initial states, and so on. For example, when only changing the process numbers of components in the Community Climate System Model Version 3 (Collins et al., 2006) (CCSM3), the result of climate simulation can be changed significantly (Song et al., 2012). This is because the CCSM3 cannot keep bit-identical simulation result in different parallel settings and the simulation result is very sensitive to the round-off errors introduced by changing the parallel setting.

To investigate how to achieve the technical reproducibility in a simple manner, we enhance the technical reproducibility on the Community Coupler (C-Coupler) platform (Liu





GMDD
7, 4429–4461, 2014



et al., 2014b). The C-Coupler is a community coupler for earth system modelling developed in China, and the C-Coupler platform is a uniform runtime environment which operates various kinds of simulations of various model versions in the same manner. Recently, the first version (C-Coupler1) was finished and publicly released with the enhancement of the technical reproducibility. Now it has been used to construct several coupled model versions by several model groups in China. The C-Coupler platform facilitates the achievement of the technical reproducibility with the following aspects. First, when configuring a model simulation with a simple command, all information for the technical reproducibility is produced and recorded into an experimental setting package automatically. Second, the data files of simulation result can contain the keywords of the corresponding experimental setting package. In other words, given a data file, scientists can reproduce the corresponding simulation. Third, given an experimental setting package, scientists can use a simple command to download the corresponding source code and input data and then prepare the environment for reproducing.

The remainder of this paper is organized as follows. Section 2 briefly introduces the C-Coupler1, including the C-Coupler platform. Section 3 analyzes the necessary information for the technical reproducibility. Section 4 details how to enhance the technical reproducibility on the C-Coupler platform. Section 5 gives out our experiences and suggestions for the technical reproducibility. They are not limited to the C-Coupler platform. We believe that they can help other model platforms to enhance the technical reproducibility. Section 6 empirically evaluates the technical reproducibility on the C-Coupler platform. We discuss and conclude this work in Sect. 7.

## 2 Brief introduction to the C-Coupler1

The C-Coupler1 is a new coupler for earth system modelling. It links component models together to construct a coupled model, achieves the parallel computation among multiple component models, controls the integration of the whole coupled model, and provides a uniform runtime environment named the C-Coupler platform which operates





GMDD
7, 4429–4461, 2014



various kinds of model simulations in the same manner. It is a parallel coupler which achieves bit-identical result with different numbers of processes. It supports three-dimensional (3-D) coupling with more flexible 3-D interpolation, and provides the functionality of integrating external algorithms to enable the same code of the C-Coupler and component models to be shared by various coupled models. Now, the C-Coupler1 has been released for public use and several coupled model versions have been constructed with it. In future, more and more model groups may use it to construct coupled models and use the C-Coupler platform to build up an internal software platform for model development. More details of the C-Coupler1 can be found in Liu et al. (2014b).

In the following context of this section, we would like to introduce the C-Coupler platform with more details. The C-Coupler platform manages the source code and input data for various model simulations. It designs the same manner for uniformly operating various model simulations on various hardware platforms. On this platform, there are four steps to operate a model simulation: "*create case*", "*configure*", "*compile*" and "*run case*". "*create case*" means creating a simulation. There are two approaches: creating the default simulation of a model version using the script "*create_newcase*" and creating a simulation from an existing simulation. The C-Coupler platform facilitates the second approach. At each time of "configure" of a simulation, the corresponding experimental setting package is automatically generated and stored. With an experimental setting package to ensure the technical reproducibility, which will be introduced later in this paper, users can reproduce an existing simulation or develop new model simulations. After a model simulation is created, users can modify the experimental setting, including the input parameters, parallel settings, hardware platform, compiling options, output settings, start and stop time, etc. After the modification of the experiment setting, users should "*configure*" the simulation and then can "*compile*" and "run case". For more information about the C-Coupler platform, please refer to its users' guide (Liu et al., 2014a).





GMDD
7, 4429–4461, 2014



## 3 Necessary information for achieving the technical reproducibility

A model simulation is actually an execution of a program on a computer system after compiling the source code and setting the specific inputs. To guarantee the technical reproducibility, scientists should make the whole environment of the model simulation able to be reproduced. Generally, the whole environment of the model simulation includes the following aspects:

1. Source code of model. The source code of model is frequently modified during the model development and scientific researches. For example, when improving existing parameterization schemes or integrating new parameterization schemes, the model code needs to be modified. Model groups can use version control systems such as Subversion (SVN, available at: http://subversion.apache.org/, last access: 24 June 2014) and GIT (available at: http://git-scm.com/, last access: 24 June 2014) to manage the code versions, for tracking the changes of the model code. However, it is possible that some model simulations use temporary code which is not a version managed by the version control system. Moreover, the scientists who are not in the model groups may not use a version control system to manage the model code.

2. Input data. The input data for a model simulation includes initial data, boundary data, forcing data, etc. Comparisons among the results of multiple simulations with different input data are frequently conducted in scientific researches. Therefore, for the technical reproducibility, the information of the input data for a simulation needs to be recorded.

3. Input parameters. To run a model simulation, scientists need to specify a set of input parameters, such as the start time and stop time of the simulation, the selected parameterization schemes, the values of parameters in the parameterization schemes, the output fields, etc. In scientific researches, scientists usually want to compare the results of multiple simulations with different input parameters,





GMDD
7, 4429–4461, 2014



such as different parameterization schemes and different parameter values for the parameterization schemes. Therefore, for the technical reproducibility, the input parameters for a simulation need to be recorded. For a coupled model, there are common input parameters for all component models and individual input parameters for each component model. All of them should be recorded.

4. Parallel setting. As pointed by Song et al. (2012), the simulation result of the same simulation may be significantly changed when only modifying the parallel setting. Therefore, for a model that cannot achieve bit-identical result with different parallel settings, the parallel setting in a simulation needs to be recorded for the technical reproducibility. For a coupled model, each component model can have an individual parallel setting. All of these parallel settings should be recorded.

5. Compiler version and compiling options. The experiences from Song et al. (2012) indicate that the model simulation result is very sensitive to round-off errors. Besides the parallel setting, the process of compiling the model code can also introduce the problem of round-off errors. The bit-wise result of floating-point calculation is very sensitive to the calculation order. For example, $a + b + c$ may be different from $c + b + a$ in bit-wise result, where $a$, $b$ and $c$ are three floating-point values. Different compiler optimizations can introduce different calculation orders and different round-off errors. Generally, compiler optimizations are determined by the compiler and compiling options. Therefore, the compiler version and compiling options for a simulation need to be recorded for the technical reproducibility. For a coupled model, each component model can have individual compiler version and compiling options for compilation. All of them should be recorded.

6. Computer system. A computer system contains a set of processors. Processors can also introduce the problem of round-off errors. First, most of modern CPUs provide out-of-order architectures which can change the order of floating-point calculations. Different kinds of CPUs may have different out-of-order architectures which result in different round-off errors. Moreover, some new processors





GMDD
7, 4429–4461, 2014



such as GPU and MIC do not use the out-of-order architectures but use in-order architectures which do not change the order of floating-point calculations. Second, the registers in processors can provide more bits for keeping floating-point values than the memory. For example, the registers for keeping double-precision floating-point values can be 80-bit wide, while double-precision values in memory are 64-bit wide. Different kinds of processors may have different numbers of registers which result in different round-off errors. Therefore, the information of processors for running a simulation needs to be recorded for the technical reproducibility. Scientists always run model simulations on a high-performance computer with a number of processors. On a heterogeneous high-performance computer with different kinds of processors, such as hybrid CPU and GPU, different deployments of the processors for a model simulation may introduce different round-off errors. In this case, the deployment of processors for a model simulation also needs to be recorded.

## 4 Enhancement of the technical reproducibility on the C-Coupler platform

Figure 1 shows the flow for achieving the technical reproducibility. It can be partitioned into two stages: recording the information of a simulation (left part in Fig. 1) and reproducing the simulation (right part in Fig. 1). In the following context of this section, we will detail the implementation on the C-Coupler platform for each stage.

### 4.1 Recording the information for the technical reproducibility

As shown in Fig. 1, the information for the technical reproducibility of a simulation is recorded in experimental setting package, log files and output data files.





GMDD
7, 4429–4461, 2014





#### 4.1.1 Experimental setting package for the technical reproducibility

When a model simulation is configured, an experimental setting package is generated automatically. This package is named as *simulation_name*.config.*configuration_time*.tar, where *simulation_name* is the name of the simulation, *configuration_time* is the time of configuring the simulation. The format of *configuration_time* is the calendar time like *YYYYMMDD-HHMMSS*. The *simulation_name* and *configuration_time* are treated as the keywords for the technical reproducibility, which are also used to name the log files and label the output data files of the simulation.

According to Sect. 3, the experiment setting package should record the information related to the model code, input data, input parameters, parallel setting, compiler version and compiling options, and computer system, as shown in Fig. 2. Moreover, the log information for configuring is recorded in the package.

**Reproducibility for the model code**

On the C-Coupler platform, the code of all component models is stored under the directory *models*. GIT is used as the default version control system for the model code and each component model can have its individual code version control system. When configuring a simulation, the code version number and the GIT sever for downloading the code of each component model are detected and recorded. It is possible that the model code for a simulation is not fully managed by GIT. For example, some code modification has not been committed as a new code version before configuring the simulation. For this case, we implemented a function called as *code patching*. When configuring a simulation, the C-Coupler platform will differentiate the corresponding model code with the related code versions managed by GIT and record the differences as code patches. When the model code is not managed by any version control system, all model code for a simulation will be recorded as code patches. Besides GIT, any other version control system such as SVN can also be used as the default version





GMDD
7, 4429–4461, 2014



control system for the model code, through slightly modifying several scripts of the C-Coupler platform.

### Reproducibility for the input data

On the C-Coupler platform, the input data files for all model simulations are put under the same directory *inputdata*, in order for promoting the sharing of input data files. SVN is used as the default version control system for tracking the changes of the input data files and for downloading these files. When configuring a simulation, all required input data files will be linked to the working directory of this simulation, and the list of these input data files is recorded. For the technical reproducibility, the SVN server, version number and checksum of each input data file will be detected and recorded. For the input data files that are not managed by SVN, only the checksum is recorded, and these files will not be put into patches of a simulation for saving the storage space.

### Reproducibility for the input parameters

On the C-Coupler platform, all files of input parameters for a simulation are generated by scripts when configuring the simulation. To modify the input parameters, scientists should modify the corresponding scripts and then configure the simulation before running it. For the technical reproducibility, all scripts for generating the files of input parameters are recorded automatically when configuring a simulation.

### Reproducibility for the parallel setting

For a simulation on the C-Coupler platform, there is a configuration file which specifies the parallel setting of every component model. The parallel setting includes the number of processes and number of threads to run the corresponding component model. After changing the parallel setting of a component model, scientists should configure the simulation and sometimes recompile the component model. When configuring a simulation, the parallel setting is recorded automatically for the technical reproducibility.





GMDD
7, 4429–4461, 2014



### Reproducibility for the compiler version and compiling options

For each component model in a simulation on the C-Coupler platform, there is a configuration file for specifying the compiler version and compiling options for compilation. After changing compiler version and compiling options, scientists should configure the simulation and recompile the corresponding component model. When configuring a simulation, for each component model, the C-Coupler platform will record the information of the corresponding compiler version, such as the name and the version number, and record the compiling options.

### Reproducibility for the computer system

On the C-Coupler platform, each computer system has a unique name, such as "Tianhe1", which was the world's fastest computer from October 2010 to June 2011. With this name, scientists can find the corresponding computer system or get more information from the web. To enable model simulations to run on a computer system, there is a configuration file which specifies how to submit a model simulation on that computer system. When configuring a model simulation, the corresponding computer system is recorded for the technical reproducibility. Currently, we only consider homogeneous computer systems where all processors are the same. Therefore, the deployment of processors for running a model simulation is not concerned yet.

### Log information for configuring

In order to record more information for the technical reproducibility, the C-Coupler platform logs the username (the user account on the computer for configuring the model simulation), computer name, configuration time, and error and warning reports for the configuration.





GMDD
7, 4429–4461, 2014



### 4.1.2 Log files for the technical reproducibility

When compiling the model code, the log information for compiling the code of each component model is recorded and the log file is named with the corresponding *simulation_name* and *configuration_time*. When running the model simulation, the log information for the execution of each component model is recorded and the log file is also named with the corresponding *simulation_name* and *configuration_time*.

### 4.1.3 Output data files for the technical reproducibility

To make the output data files of a simulation able to be reproduced, the C-Coupler1 as well as the C-Coupler platform provides a user application interface (API) *c_coupler_log_case_info_in_netcdf_file*. Using this API, a component model will write a series of information into the output data files for the simulation, including the *simulation_name*, *configuration_time*, the name of the corresponding model version, and the description of the simulation. Given an output data file, scientists can find the corresponding experimental setting package and log files according to the *simulation_name* and *configuration_time*, and then the simulation corresponding to the output data file can be reproduced. We advise scientists to leave their contact information in the description of the model simulation, such as email address, so that other scientists can easily contact the scientists who originally run the simulation. The API *c_coupler_log_case_info_in_netcdf_file* only supports the output data file in NETCDF format. In future, we will provide similar APIs to support more formats of output data files.

### 4.1.4 Operations for recording the information for the technical reproducibility

An experimental setting package is generated automatically when scientists configure a simulation on the C-Coupler platform. As mentioned in Sect. 4.1.2, the checksum of each input data file needs to be calculated for the technical reproducibility. Considering





GMDD
7, 4429–4461, 2014



that it is time-consuming to calculate the checksum of a data file, especially when the size of the file is large, we provide two types of configuration. When scientists use the command "*configure*", the checksums of the input data files are neglected, and when scientists use the command "*configure -checksum*", the checksums are calculated and recorded. To achieve the technical reproducibility, scientists should take the following operations for a simulation on the C-Coupler platform:

1. Use the command "*configure -checksum*" to configure the simulation, after modifying the input parameters, parallel setting, compiling options, etc.;
2. Use the command "*compile*" to compile the model code;
3. Use the command "*runcase*" to run the model simulation.

### 4.2 Reproducing a model simulation

To reproduce a simulation, scientists should find out the corresponding experimental setting package. Scientists are suggested to preserve the experimental setting packages for various simulations. Then given a data file of simulation result, the corresponding experimental setting package can be found. Generally, the size of an experimental setting package is very small (e.g., smaller than 1 MB), especially when the model code and input data files are well managed by version control systems. Therefore, it is convenient to keep a large number of experimental setting packages on modern storage. With an experimental setting package, scientists can reproduce the corresponding simulation with two major steps: downloading the source code and input data files for the simulation and then repeating it.

#### 4.2.1 Downloading for a model simulation

As introduced in Sect. 4.1, most of necessary information for the technical reproducibility has been recorded in the experimental setting package, except the model code and





GMDD
7, 4429–4461, 2014



input data files, which need to be downloaded when reproducing a simulation. Therefore, a script named *checkout_experiment* is implemented for downloading the model code and input data files. The command for downloading is "*checkout_experiment experimental_setting.tar local_env*", where *experimental_setting.tar* is an experimental setting package and *local_env* specifies several environmental variables, such as the directories for storing the model code and input data files.

For downloading the model code, the script *checkout_experiment* first checkouts the corresponding model code version of each component model, and then recover the code patches that are recorded in the experimental setting package. For downloading the input data files, the script *checkout_experiment* iterates on each input data file. If a data file already exists in the corresponding directory specified by the file *local_env* and is the same as the wanted (i.e., the checksum of the file is the same as that recorded in the experimental setting package), the data file will not be downloaded anymore. Otherwise, it will be downloaded from the corresponding SVN server. It is possible that there is no SVN server for downloading the input data files. In this case, the script *checkout_experiment* can help verify whether the local input data files are the same as the wanted.

### 4.2.2 Repeating a model simulation

After downloading and verifying the model code and input data files for a simulation, scientists can take the following steps to repeat the simulation and reproduce the simulation result.

1. Create a working directory for the simulation and unpack the corresponding experimental setting package under this directory.

2. Link the simulation to the directories for the downloaded model code and input data files, through setting environmental variables.





GMDD
7, 4429–4461, 2014



3. The default parallel setting is the same as that recorded in the experimental setting package. If the corresponding model version can achieve bit-identical result with various parallel settings, the parallel setting can be modified flexibly.

4. Verify the compiler version and install the corresponding compiler version if necessary. Section 5 will discuss about which compiler versions can help achieve bit-identical result.

5. Verify the computer system. Section 5 will discuss about which processor versions can help achieve bit-identical result.

6. Configure, compile, and run the model simulation.

7. Check whether the bit-identical simulation result is reproduced. The log files, output data files and figures from the original model simulation can help this check.

8. If the bit-identical simulation result is not reproduced, please contact the authors of the original model simulation. One possible cause for this situation is that there are bugs in the model code.

## 5 Experiences and suggestions for the technical reproducibility

In this section, we would like to share our experiences regarding to the technical reproducibility, including how to make the parallelization of models achieve bit-identical simulation result in different parallel settings, and which compiler versions and processor versions can produce bit-identical simulation result. We call the parallelization that achieves bit-identical result in different parallel settings as bit-identical parallelization, the set of compiler versions that achieve bit-identical result as bit-identical compiler version set, and the set of processor versions that achieve bit-identical result as bit-identical processor version set. Moreover, we would like to give some suggestions for facilitating the achievement of the technical reproducibility.





GMDD
7, 4429–4461, 2014



### 5.1 Experiences regarding to the technical reproducibility

#### 5.1.1 Experiences for the bit-identical parallelization

The C-Coupler platform does not require models to achieve bit-identical parallelization, because the parallel settings of a model simulation will be recorded in the experimental setting package for the technical reproducibility. However, we highly recommend the bit-identical parallelization of models due to two reasons. First, it is a strict and valuable testing standard for correct parallelization. Generally, wrong parallelization cannot guarantee bit-identical result with different parallel settings. If a parallel version is allowed to produce different results when the parallel setting is modified, it will be difficult to verify whether the parallel version is right or not. Second, when the bit-identical parallelization is achieved, scientists can freely change the parallel settings when reproducing a model simulation. If the computing resource is limited, scientists can use a smaller number of processor cores to reproduce bit-identical simulation result.

For achieving the bit-identical parallelization, we suggest scientists pay attention to the following two aspects and consider the corresponding suggestions:

1. Compiling options. Different compiler optimizations may lead to different calculation orders of floating-point operations, different round-off errors, and different simulation results. We find that, the bit-identical parallelization is relevant to compiler optimizations, and the optimization level O0, which keeps the original calculation orders in the model code, is more helpful for achieving the bit-identical parallelization. For higher optimization levels, scientists are advised to find out the compiling options which guarantee the original calculation orders. For example, regarding the Intel compiler, the set of compiling options "-no-vec -mp1 -fp-model precise -fp-speculation=safe" can guarantee the original calculation orders at the optimization level higher than O0. This set of compiling options is highly recommended for model development when using the Intel compiler. Although these compiling options may somehow restrict the compiler optimizations, the





GMDD
7, 4429–4461, 2014



decrement of the computational performance is very limited, which is observed in our simulations.

2. Global summation operations. There are always global summation operations in the model code. Scientists can use the reduce function in the Message Passing Interfaces (MPI) library to achieve global summation in parallel. However, this implementation possibly results in non-bit-identical parallelization because the MPI reduce function will change the order of floating-point calculations and introduce different round-off errors when using different parallel settings. To achieve bit-identical global summation, we propose two approaches. First, one process can be selected to gather all data values and calculate the sum, and then broadcasts the sum to all processes if required. Second, the MPI reduce function can be used to calculate summation with higher precision than that of the corresponding floating-point values. For example, for single-precision floating-point values, double-precision summation should be used, and for double-precision values, long-double-precision (128-bit wide) summation should be used. Although higher-precision summation may introduce higher computation cost, especially for the long-double-precision summation, we prefer the second approach because it performs summation in parallel and the corresponding communication cost is much smaller than that of the first approach.

### 5.1.2 Experiences on bit-identical compiler version set and processor version set

It is unnecessary to use exactly the same compiler version and processor version for reproducing the bit-identical result of a model simulation. For example, in our model group, we use the Intel Fortran and C/C++ compiler versions and Intel CPU versions for model development. We find that, if the compiling options "-no-vec -mp1 -fp-model precise -fp-speculation=safe" are used, a model simulation can achieve bit-identical result no matter which Intel compiler version is selected from *Version 12.1.3, 13.0.0 or 14.0.2*,





GMDD
7, 4429–4461, 2014



and which Intel CPU version is selected from *Xeon E5520, X5670 or E5-2670*. In other words, the Intel compiler versions *12.1.3, 13.0.0 and 14.0.2* constitute a bit-identical compiler version set, and the Intel CPU versions *Xeon E5520, X5670 and E5-2670* constitute a bit-identical processor version set.

## 5.2 Suggestions for facilitating the achievement of the technical reproducibility

To facilitate the achievement of technical reproducibility, we will give a series of suggestions from different aspects, including model code, input data, compiler, processor, experimental setting package and bit-identical testing.

### 5.2.1 Suggestions for model code management

Model groups usually use a version control system to track the development of model code. Currently, the popular version control systems include SVN and GIT. SVN or GIT can be used as the default version control system for model code on the C-Coupler platform. We prefer GIT rather than SVN because GIT is a distributed version control system, which can facilitate multiple model groups to cooperatively develop a coupled model. Moreover, GIT provides more functions for version control, such as branch version, local commits, etc.

### 5.2.2 Suggestions for input data management

When model code is not managed by a version control system, the technical reproducibility can also be achieved due to the implementation of the code patching function on the C-Coupler platform. We do not implement a function of *data patching* because the total size of input data files is large. When the input data files of a model simulation are not managed by a version control system, they must have been already prepared on the local computer when reproducing the model simulation, because they cannot be downloaded from a public server. We therefore recommend scientists to use a version control system to manage input data files. On the C-Coupler platform, the default





GMDD
7, 4429–4461, 2014



version control system for input data files is SVN but not GIT because SVN is friendlier for downloading data files.

Besides the facilitation for the technical reproducibility, more information can be provided by the version control of input data files for scientific researches. For example, the log information for tracking a change can record where the corresponding input data files are downloaded from, who downloads or generates these input data files, and how to generate these input data files. In a model group, the input data files under version control can be shared by all scientists, so as to save storage space.

### 5.2.3 Suggestions for compilers and processors

When reproducing a model simulation, the compiler and processor should be concerned. If the bit-identical result of a model simulation highly depend on a specific compiler version and a specific processor version, it will be inconvenient for scientists to reproduce the model simulation. For example, scientists may have to find a computer with the specific processor version and install the specific compiler version on that computer, which will introduce a lot of work to do. We therefore give the following suggestions:

1. Use common compilers and common processors for the model development. Common processors such as the Intel CPUs have been used worldwide, which makes it easy to find a computer for reproducing. Common compilers such as the GNU compilers and Intel compilers have often been installed on computers, which can possibly save the work for installing the compiler version for reproducing.

2. Extend the bit-identical compiler version set and processor version set. As introduced in Sect. 5.1.2, multiple compiler versions can constitute a bit-identical compiler version set and multiple processor versions can constitute a bit-identical processor version set. Through extending these two sets, it will become more convenient to reproduce model simulations. These two sets discovered by us currently are limited to the Intel Compiler versions and Intel CPU versions. In future,





GMDD
7, 4429–4461, 2014



we will try to extend these two sets through investigating and including more compilers and more processors, such as the GNU compiler versions and the AMD CPU versions.

### 5.2.4 Suggestions for experimental setting package

When configuring a model simulation, the corresponding experimental setting package is generated automatically. There are two choices for configuring: command "*configure*" and command "*configure -checksum*" (Sect. 4.1.4). The command "*configure -checksum*" is designed and implemented for the technical reproducibility. It queries the SVN server and version number (if the input data files are managed by SVN) and calculates the checksum of each input data file. We therefore advise scientists to use this command for important simulations.

Given an output data file, the first step for reproducing the corresponding simulation is to find out the corresponding experimental setting package, according to the log information in the output data file. We therefore suggest scientists to well keep the experimental setting packages, especially for important simulations. For a model group, there could be a shared directory to store all experimental setting packages provided by different scientists. The version control systems such as SVN and GIT and data bases can help manage the experimental setting packages. As the size of an experimental setting package is small, a large number of experimental setting packages can be kept in a long time.

### 5.2.5 Suggestions for bit-identical testing

The testing of model programs, which targets to detecting and fixing bugs, is very important for model development. The achieving of the technical reproducibility requires scientists to strictly test the model programs. Moreover, it opens new testing opportunities for model development. In the following, we propose a list of bit-identical testing standards for reference:





GMDD
7, 4429–4461, 2014



1. Bit-identical parallelization. Bit-identical parallelization, which requires a model simulation to achieve bit-identical result in different parallel settings, can improve the flexibility of reproducing a mode simulation. It also provides a strict testing standard for detection and correction of bugs in parallelization.

2. Bit-identical result of a simulation in repeated runs. The technical reproducibility requires a model simulation to achieve bit-identical result when repeating the same runs for any number of times under the same environment. If the repeated runs do not achieve bit-identical result, it is very likely that there are bugs in the corresponding model code, such as uninitialized variables and out-of-bounds array accesses. Following the development of physical parameterization schemes, random numbers may be introduced to the descriptions of some stochastic processes in the schemes. In this case, to guarantee the technical reproducibility and to advance the testing of model code, the random numbers must be able to be reproduced. One choice is to use pseudorandom numbers, a deterministic sequence of numbers that approximate the properties of random numbers. Pseudorandom numbers are determined by a small set of initial values. When the initial values are treated as input parameters, the pseudorandom numbers can be fully reproduced.

3. Bit-identical result of a simulation in initial run and restarted runs. CSMs and ESMs are required to take a long time, such as several months, to simulate the variation of climate in hundreds even thousands of simulation years. The restart function is essential to guarantee the success of long-time simulation. A correct restarted run should keep bit-identical result with the corresponding initial run. For example, given a five-year initial run and a corresponding restarted run (i.e., end the initial run at the 2nd simulation year and then continue the simulation through restarting) of the same model simulation, these two runs should achieve bit-identical result. If not, there are potentially bugs in the model code.





GMDD
7, 4429–4461, 2014



4. Bit-identical result of a simulation on different computer environments. Given a bit-identical compiler version set with $N$ compiler versions and a bit-identical processor version set with $M$ processor versions, there are $M \cdot N$ selections of computer environments. If a model simulation cannot achieve bit-identical result on the $M \cdot N$ computer environments, it is very likely that there are bugs in the corresponding model code, such as uninitialized variables and out-of-bounds array accesses.

5. More and more simulations accumulated for testing. We call the above three suggestions as bit-identical testing standards. It is easy to verify whether a model version passes bit-identical tests. We therefore suggest model groups accumulate more and more simulations for testing various model versions automatically.

## 6 Empirical evaluation

To empirically evaluate the enhancement of the technical reproducibility, we use two coupled model versions that are already on the C-Coupler platform, e.g., the FGOALS-gc and MASNUM-POM. The FGOALS-gc is a modified version based on the FGOALS-g2 (Li et al., 2013a), a CMIP5 CSM, where the original coupler CPL6 (Craig et al., 2005) used in the FGOALS-g2 is replaced by the C-Coupler1. The MASNUM-POM is a coupled model version consisting of a wave model MASNUM (Yang et al., 2005) and a parallel version of the ocean model POM (Wang et al., 2010).

In this evaluation, we use three different computer platforms. Table 1 shows the processor version and Operating System (OS) version on each computer platform. Although all of these three computer platforms use the Intel CPU and Linux OS, the versions of the processor and OS are different. As introduced in Sect. 5.1.2, the Intel CPU versions on these three computer platforms belong to the same bit-identical processor version set. On each computer platform, we installed three versions of the Intel compiler, including Version 12.1.3, 13.0.0 and 14.0.2, which belong to the same bit-identical





GMDD
7, 4429–4461, 2014



compiler version set. In each compilation of the model code, we add "-no-vec-mp1-fp-model precise -fp-speculation=safe" to the corresponding compiling options.

Corresponding to each coupled model version, one simulation is employed for this evaluation. Given a simulation, we conduct the empirical evaluation with the following steps:

1. Establish a benchmark simulation run on the computer platform No. 1 using the Intel compiler Version 12.1.3. The corresponding experimental setting package is generated.

2. Use the experimental setting package to reproduce the simulation on the computer platform No. 1 using each other Intel compiler version, e.g., Version 13.0.0 and 14.0.2.

3. On each other computer platform, e.g., No. 2 and No. 3, use the experimental setting package to download the simulation from the computer platform No. 1, and then reproduce the simulation using each Intel compiler version.

4. Check whether all reproduced simulation runs achieve the bit-identical result with the benchmark simulation run. There are totally 8 reproduced simulation runs.

The evaluation result shows that, for the simulation of each coupled model version, the 8 reproduced simulation runs achieve the bit-identical result with the benchmark simulation run. We therefore can conclude that the C-Coupler platform can achieve the technical reproducibility for model simulations.

## 7 Discussion and conclusion

In this paper, we focus on how to achieve the technical reproducibility. After analyzing the necessary information for it, we present how to enhance it on the C-Coupler platform and give a series of experiences and suggestions for achieving it. Now we can





GMDD
7, 4429–4461, 2014



conclude that the technical reproducibility of model simulations is achievable and is prospective to advance earth system modelling in future.

This work focuses on the technical reproducibility for the execution of models. It can be extended to achieve the technical reproducibility for pre-process and post-process of models. In future work, we will integrate the pre-process and post-process into the C-Coupler platform, with the enhancement of the technical reproducibility for them.

The C-Coupler platform is a new software platform for model development. There are also other software platforms which have been successfully and widely used for model development, e.g., the CCSM3 platform (Collins et al., 2006), the CCSM4/CESM platform (Gent et al., 2011; Hurrell et al., 2013), the FMS (Balaji, 2004), etc. We believe that the design and implementation for the technical reproducibility on the C-Coupler platform can be easily extended to other platforms. Moreover, we give a series of experiences and suggestions for the technical reproducibility. These experiences and suggestions are originated from the model development in our model group. We believe that they can benefit the model development in other model groups.

In our model development, the technical reproducibility is very effective for testing models. It can dramatically improve the cooperation, knowledge sharing and spreading among scientists. We believe that the technical reproducibility deserves to be a worldwide standard for the development of earth system modelling. For example, in future, various MIPs and scientific publications should encourage the participants or authors to provide easy reproduction of each model simulation, e.g., submitting the corresponding experimental setting package, making the corresponding model platform, model code and input data files able to be downloaded, etc.

In our works to achieve the technical reproducibility, we use the *simulation_name* and *configuration_time* as the keywords for the technical reproducibility and record them in the output files of a model simulation. As a result, the corresponding model simulation can be reproduced according to an output file. The analysis of model simulation result is generally based on the figures generated by diagnostic packages. The total size of the figures is potentially much smaller than that of the corresponding output data files,





GMDD
7, 4429–4461, 2014



especially when the resolutions of models get very high. We therefore prefer to keep the figures rather than the output data files to save storage space, especially when the corresponding model simulation is not significant enough. This requires that the corresponding model simulation able to be reproduced according to a figure. A simple solution for this requirement is to put all figures of a model simulation into a directory named with the *simulation_name* and *configuration_time*. A better solution is to log these keywords in the file of each figure, which requires the modification of the diagnostic packages.

The facilitation of the technical reproducibility also requires the efforts beyond the field of earth system modelling. Through enlarging the bit-identical compiler version set and processor version set, the bit-identical result of a model simulation can be reproduced more easily and flexibly. We therefore hope the experts in the field of computer will make the new versions of compilers and processors join in existing bit-identical compiler version sets and processor version sets. For example, we find that, the model simulation result with the Intel compiler version 11.1 cannot be reproduced when using a newer version of the Intel compiler, e.g., Version 12.1.3, 13.0.0 and 14.0.2. This could introduce trouble to model development. When a model group wants to advance the Intel compiler version from 11.1 to 12.1.3, existing simulations are expected to be rerun and reevaluated, which introduces a lot of extra work. If all future Intel compiler versions will join in the bit-identical compiler version set consisting of the Version 12.1.3, 13.0.0 and 14.0.2, the current simulations will be able to be reproduced with the latest compiler version in future and model groups will be free to advance compiler versions.

*Acknowledgements.* This work is supported in part by the Natural Science Foundation of China (no. 41275098), the National Grand Fundamental Research 973 Program of China (no. 2014CB441302) and the Tsinghua University Initiative Scientific Research Program (no. 20131089356).





GMDD
7, 4429–4461, 2014



## References


Balaji, V.: FMS: the GFDL Flexible Modelling System, available at: http://www.gfdl.noaa.gov/~fms/ (last access: 24 June 2014), 2004.

Booij, N., Ris, R. C., and Holthuijsen, L. H.: A third-generation wave model for coastal regions: 1. model description and validation, J. Geophys. Res., 104, 7649–7666, 1999.

Craig, A. P., Jacob, R. L., Kauffman, B., Bettge, T., Larson, J. W., Ong, E. T., Ding, C. H. Q., and He, Y.: CPL6: the new extensible, high performance parallel coupler for the Community Climate System Model, Int. J. High Perform. C., 19, 309–327, 2005.

Collins, W. D., Bitz, C. M., Blackmon, M. L., Bonan, G. B., Bretherton, C. S., Carton, J. A., Chang, P., Doney, S. C., Hack, J. J., Henderson, T. B., Kiehl, J. T., Large, W. G., McKenna, D. S., Santer, B. D., and Smith, R. D.: The Community Climate System Model version 3 (CCSM3), J. Climate, 19, 2122–2143, 2006.

Dennis, J., Edwards, J., Evans, K. J., Guba, O., Lauritzen, P. H., Mirin, A. A., St-Cyr, A., Taylor, M. A., and Worley, P. H.: CAM-SE: a scalable spectral element dynamical core for the Community Atmosphere Model, Int. J. High Perform. C., 26, 74–89, 2012.

Gent, P. R., Danabasoglu, G., Donner, L. J., Holland, M. M., Hunke, E. C., Jayne, S. R., Lawrence, D. M., Neale, R. B., Rasch, P. J., Vertenstein, M., Worley, P. H., Yang, Z.-L., and Zhang, M.: The Community Climate System Model version 4, J. Climate, 24, 4973–4991, 2011.

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

Griffies, S. M., Schmidt, M., and Herzfeld, M.: Elements of MOM4p1, GFDL Ocean Group Technical Report 6, Geophysical Fluid Dynamics Laboratory, Princeton, 2010.

Hunke, E. C. and Lipscomb, W. H.: CICE: the Los Alamos Sea Ice Model, documentation and software, version 4.0, Los Alamos National Laboratory Tech. Rep. LA-CC-06-012, Los Alamos National Laboratory, Los Alamos, NM, 2008.






GMDD
7, 4429–4461, 2014



Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque, J.-F., Large, W. G., Lawrence, D., Lindsay, K., Lipscomb, W. H., Long, M. C., Mahowald, N., Marsh, D. R., Neale, R. B., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., Collins, W. D., Hack, J. J., Kiehl, J., and Marshall, S.: The Community Earth System Model: a framework for collaborative research, B. Am. Meteorol. Soc., 94, 1339–1360, 2013.

Li, L. J., Lin, P. F., Yu, Y. Q., Wang, B., Zhou, T. J., Liu, L., Liu, J. P., Bao, Q., Xu, S. M., Huang, W. Y., Xia, K., Pu, Y., Dong, L., Shen, S., Liu, Y. M., Hu, N., Liu, M. M., Sun, W. Q., Shi, X. J., Zheng, W. P., Wu, B., Song, M.-R., Liu, H. L., Zhang, X. H., Wu, G. X., Xue, W., Huang, X. M., Yang, G. W., Song, Z. Y., and Qiao, F. L.: The Flexible Global Ocean–Atmosphere–Land System Model: grid-point version 2: FGOALS-g2, Adv. Atmos. Sci., 30, 543–560, 2013a.

Li, L. J., Wang, B., Dong, L., Liu, L., Shen, S., Hu, N., Sun, W., Wang, Y., Huang, W., Shi, X., Pu, Y., and Yang., G.: Evaluation of Grid-point Atmospheric Model of IAP LASG version 2 (GAMIL2), Adv. Atmos. Sci., 30, 855–867, doi:10.1007/s00376-013-2157-5, 2013b.

Lipscomb, W., Bindschadler, R., Bueler, E., Holland, D., Johnson, J., and Price, S.: A community ice sheet model for sea level prediction: building a next-generation Community Ice Sheet Model, EOS T. Am. Geophys. Un., 90, 23–23, 2009.

Liu, H. L., Lin, P. F., Yu, Y. Q., and Zhang, X. H.: The baseline evaluation of LASG/IAP Climate system Ocean Model (LICOM) version 2.0, Acta Meteorol. Sin., 26, 318–329, 2012.

Liu, L., Li, R., Zhang, C., Yang, G., and Wang, B.: Community coupler C-Coupler1 User's Guide, available at: http://www.cess.tsinghua.edu.cn/publish/ess/7687/20120515135108504806682/CommunityCouplerC-Coupler1UserGuide.pdf (last access: 24 June 2014), 2014a.

Liu, L., Yang, G., Wang, B., Zhang, C., Li, R., Zhang, Z., Ji, Y., and Wang, L.: C-Coupler1: a Chinese community coupler for Earth System Modelling, Geosci. Model Dev. Discuss., 7, 3889–3936, doi:10.5194/gmdd-7-3889-2014, 2014b.

Ji, D. and Dai, Y.: The Common Land Model (CoLM) Technical Guide, available at: http://globalchange.bnu.edu.cn/download/doc/CoLM/CoLM_Technical_Guide.pdf (last access: 24 June 2014), 2008.

Oleson, K. W., Dai, Y., Bonan, G. B., Bosilovichm, M., Dickinson, R., Dirmeyer, P., Hoffman, F., Houser, P., Levis, S., Niu, G.-Y., Thornton, P., Vertenstein, M., Yang, Z., and Zeng, X.: Technical Description of the Community Land Model (CLM), NTIS #PB2004-105836, 2004.





GMDD
7, 4429–4461, 2014



Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D O., Barker, D. M., Wang, W., and Powers, J. G.: A Description of the Advanced Research WRF Version 2, NCAR Tech, 2005.

Smith, R. D. and Gent, P. R.: Reference manual for the Parallel Ocean Program (POP), ocean component of the Community Climate System Model (CCSM2.0 and 3.0), Technical Report LA-UR-02-2484, Los Alamos National Laboratory, available at: http://www.ccsm.ucar.edu/models/ccsm3.0/pop (last access: 24 June 2014), 2002.

Song, Z., Qiao, F., Lei, X., and Wang, C.: Influence of parallel computational uncertainty on simulations of the Coupled General Climate Model, Geosci. Model Dev., 5, 313–319, doi:10.5194/gmd-5-313-2012, 2012.

Wang, G., Qiao, F., and Xia, C.: Parallelization of a coupled wave-circulation model and its application, Ocean Dynam., 60, 331–339, 2010.

Yang, Y., Qiao, F., Zhao, W., Teng, Y., and Yuan, Y.: MASNUM ocean wave numerical model in spherical coordinates and its application, Acta Oceanol. Sin., 27, 1–7, 2005.





GMDD

7, 4429–4461, 2014



**Table 1.** Three computer platforms used in the empirical evaluation.

| No. | Processor version | Number of cores of each CPU | Operating System (OS) version |
|---|---|---|---|
| 1 | Intel Xeon E5520 | 4 | Linux 2.6.18-371.1.2.el5 |
| 2 | Intel Xeon X5670 | 6 | Linux 2.6.18-194.17.1.0.1.el5_lustre.1.8.5 |
| 3 | Intel Xeon E5-2670 | 8 | Linux 2.6.32-431.11.2.el6.x86_64 |





GMDD
7, 4429–4461, 2014



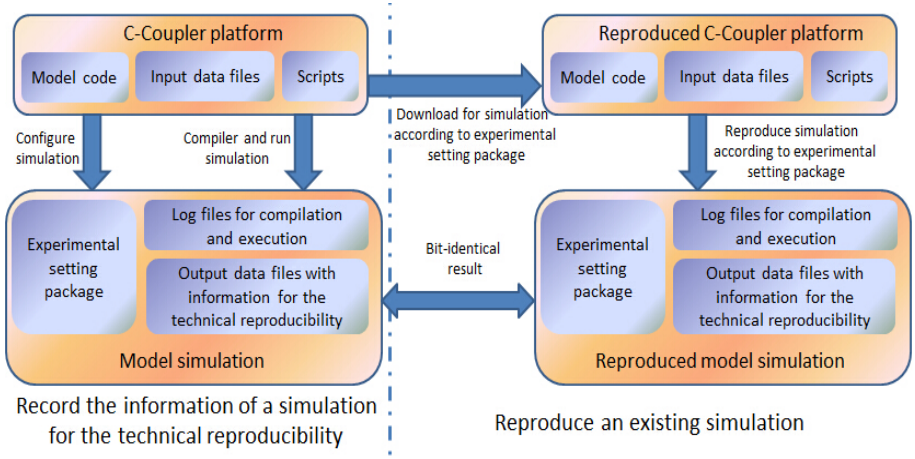


**Figure 1.** Flowchart for achieving the technical reproducibility on the C-Coupler platform.





GMDD
7, 4429–4461, 2014



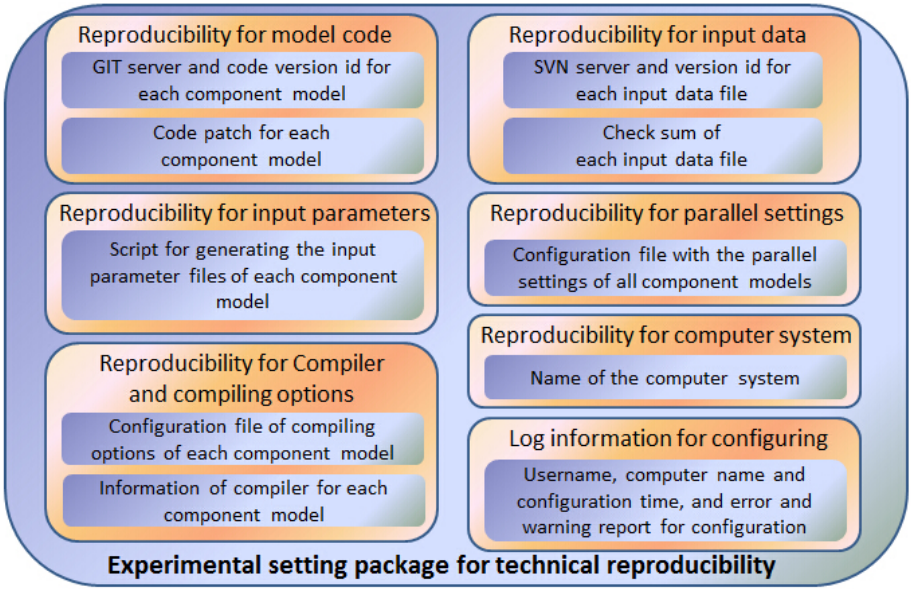


**Figure 2.** Architecture of the experimental setting package for the technical reproducibility.





GMDD
7, 4429–4461, 2014