Geosci. Model Dev. Discuss., 7, C1088–C1091, 2014
www.geosci-model-dev-discuss.net/7/C1088/2014/

Geoscientific
Model Development
Discussions
Open Access

Interactive comment on “C-Coupler1: a Chinese community coupler for Earth System Modelling” *by* L. Liu et al.

L. Liu et al.

liuli-cess@tsinghua.edu.cn

Received and published: 10 July 2014

We thank Reviewer #2 for the comments, suggestions, and the detailed modifications for the grammatical errors. We will modify the manuscript according to all of them in the revision process. In the following, we select some comments to reply.

1. The scientific reproducibility I have rated fair. I think this aspect of the paper could be improved in a revision where the overall workflow of C-Coupler1 is described including an explanation of the interaction of the configuration files, API and, runtime system. For example, a scenario could be described in which a model developer takes an existing model code and prepares it for use in the C-Coupler platform, including all phases. What steps are required?

Response: We highly concern about the reproducibility of numerical simulation results not only for the C-Coupler but also for various experiment models. As briefly introduced on page 26 line 3, we enhanced the reproducibility of numeral simulation result on the C-Coupler platform. There will be a corresponding GMDD paper online soon. In the current version of this manuscript, we only focus on the description of the design and functionality of the C-Coupler1. We will add more detailed implementations according the reviewer’s suggestion in the revised version.

2. Section 1 identifies four key requirements for coupling future ESMs, including 3-D coupling, high-level sharing, e.g., some model codes could have compiler directives (#ifdef) for different configurations (requiring a recompile) while others could read a file at runtime and be configured dynamically.

Response: We will further clarify the “high-level sharing” in the revised manuscript. We do not like the use of compiler directives such as #ifdef to specify the different configurations of the same component model, because the compiler directives always make the code of model hard to be read or be further developed. Existing couplers such as OASIS can help achieve high-level sharing, but not in all cases. For example, for the standalone version of the AGCM GAMIL, the flux computation algorithm only considers the variation of SST while the flux computation algorithm (the algorithm used in the CPL6) in the CSM FGOALS-g2 further considers the variation of SSH and ocean current velocity. As OASIS cannot dynamically select the flux algorithm, code changes or compiler directives are required for the different configurations of the GAMIL.

3. The term “coupled model version” is used throughout the manuscript and could be confused with software release versions (e.g., CAM4 vs CAM5) . . . a model code is used in multiple configurations.

Response: We will clarify the term “coupled model version”. In our mind, the “coupled model version” means “experiment model”. We should move the definition of the “experiment model” into even before Section 1.

4. The definition of “common model software platform” is unclear in this section. The idea of running multiple coupled models in the same environment is understood conceptually as a way to promote interoperability, although it is not clear what it means that “various kinds of model simulations [are run] in a same manner.” I suggest adding more technical details.

Response: Here we’d like to set an example. Given that a scientific research needs to run different experiment models including the WRF, CESM and FGOALS-gc. Scientists have to learn how to configure, compile and run the WRF, CESM and FGOALS-gc respectively. We hope these models can be operated on the same platform, which means the operation for configuring, compiling and running all experiment models is the same.

5. Section 2 covers existing couplers adequately and, to my knowledge, most of the relevant coupling technologies are listed there. I suggest adding a section on the Bespoke Framework Generator (BFG) (which is already referenced in the paper) including details of the configuration metadata for that system (see the process Define, Configure, Compose, Deploy). One reason to do this is because C-Coupler1 requires a substantial amount of configuration and a comparison of the way configuration is done in both systems would better situate C-Coupler1 in the context of existing work.

Response: We will add a sub section to introduce the BFG.

6. Section 3.1 introduces the term “experiment model” as “a model version which can run on the C-Coupler platform.” This implies some compliance rules must be satisfied, but the compliance rules are not listed. Are certain configuration files required? Certain API calls? What steps are required to take an existing non-compliant model and make it run on the C-Coupler platform?

Response: Here “a model version” should be modified as “a model code version”. The same experiment model can have different compliance rules to produce different simulation results.

7. In section 3.3.1 and later in section 4.2, several kinds of configuration files are mentioned, although it took several readings to understand the difference . . . output field, algorithms, and interfaces, etc. but it not actually specifying a particular configuration. Likewise, the “configuration files of an experiment model” is similar to the BFG “Composition” phase . . . where they apply in the overall workflow would greatly clarify the system description.

Response: The configuration files in the C-Coupler describe the meta data for coupling, similar to the BFG. Like a model “Definition” in the BFG, the configurations files in the C-Coupler1 describes the input fields, output field, algorithms of a component model. The detailed implementation of these descriptions in the C-Coupler1 is different from the BFG. Similarly, the detailed implementation of the description for a coupled system is different from the BFG. We will clarify them when revising the manuscript.

8. The last sentence of section 4.1.1 says that algorithms registered through the API “do not have parameters and return value[s].” In this case, how does the algorithm find its input data (and what is the nature of that data; is it model fields?) and where does the output go?

Response: The input and output of the private external algorithms are set implicitly in the algorithm code (always Fortran code) through using modules’ public variables. In the future updated version of the C-Coupler, we may enable the private external algorithms to have explicit input and output variables.

Interactive comment on Geosci. Model Dev. Discuss., 7, 3889, 2014.