Geosci. Model Dev. Discuss., 8, 351–379, 2015
www.geosci-model-dev-discuss.net/8/351/2015/
doi:10.5194/gmdd-8-351-2015
© Author(s) 2015. CC Attribution 3.0 License.

Geoscientific Model Development Discussions Open Access

The software architecture of climate models: a graphical comparison of CMIP5 and EMICAR5 configurations

K. Alexander[1,*] **and S. M. Easterbrook**[1]

[1]Department of Computer Science, University of Toronto, 10, King's College Rd, Toronto, Ontario, Canada
[*]now at: Climate Change Research Centre and ARC Centre of Excellence for Climate System Science, University of New South Wales, Sydney NSW 2052, Australia

Received: 19 November 2014 – Accepted: 19 December 2014 – Published: 21 January 2015

Correspondence to: S. M. Easterbrook (sme@cs.toronto.edu)

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

Abstract

We analyse the source code of eight coupled climate models, selected from those that participated in the CMIP5 (Taylor et al., 2012) or EMICAR5 (Eby et al., 2013; Zickfeld et al., 2013) intercomparison projects. For each model, we sort the preprocessed code into components and subcomponents based on dependency structure. We then create software architecture diagrams which show the relative sizes of these components/subcomponents and the flow of data between them. The diagrams also illustrate several major classes of climate model design; the distribution of complexity between components, which depends on historical development paths as well as the conscious goals of each institution; and the sharing of components between different modelling groups. These diagrams offer insights into the similarities and differences between models, and have the potential to be useful tools for communication between scientists, scientific institutions, and the public.

1 Introduction

Global climate models are large and complex software systems, consisting of hundreds of thousands of lines of code, and a development history spanning years or even decades. Understanding what each model does and how it differs from other models is a difficult problem. Existing approaches to model comparison focus on measures of a model's skill in reproducing observed climates of the past, and on informal discussion of differences in how physical processes are resolved or parameterized within each model.

In this paper, we take a different approach. We characterize the software architecture of each model by analysing how the physical domains of the Earth system are modularized in the models, how these modules interact, and the relative sizes of these modules. The analysis reveals differences between models, both in terms of the architectural decisions regarding coupling between Earth system components, and also in terms of where the bulk of the code lies. We argue that these differences in module size offer a reasonable proxy for scientific complexity of each component. This in turn offers preliminary evidence that when modelling groups tend to specialize in different parts of the Earth system, these specializations are reflected in the architecture of their models.

2 Background

Intercomparison of models is now standard practice in Earth system modelling, as it provides insights into the strengths and weaknesses of each model, and generates standard model runs for more formal measurements of model skill. The World Climate Research Program website (2014) currently lists 45 active Model Intercomparison Projects (MIPs). Typically, these intercomparison projects proceed by defining an agreed set of model experiments which represent the different conditions models might be expected to simulate, often with (re-gridded) observational data provided as a baseline for comparison. Some of these intercomparison projects were also designed to provide a coordinated set of Earth system model runs as input to the IPCC assessment reports. In the 5th IPCC assessment report (AR5), the long term projections of future climate change were generated from CMIP5 (Taylor et al., 2012) for Global Coupled Climate Models (GCMs), and EMICAR5 (Eby et al., 2013; Zickfeld et al., 2013) for Earth System Models of Intermediate Complexity (EMICs).

Comparisons between models are normally expressed in terms of model skill relative to the given observational data, with skills scores computed by measuring mean-squared error for selected fields. For example, Reichler and Kim (2008) sum the mean squared errors at each grid point for each of 14 annually averaged variables, normalize them to account for grid variations in mass and area, and combine these to produce a single skill score for each model. Their results indicate that model error is steadily declining over successive generations of global climate models.

An alternative approach is to directly compare the climatology of the models against each other, by analyzing the spatial and temporal patterns simulated for a specific variable. For example, Masson and Knutti (2011) use this approach on monthly fields for surface temperature and precipitation to generate a cluster analysis on families of models. Their results show that models from the same lab tend to have similar climatology, even across model generations, as do models from different labs that use the same atmosphere or ocean components.

Understanding the relationships between different models is particularly important for creating model ensembles and probabilistic forecasts (Collins, 2007). Currently, model ensembles tend to be "ensembles of opportunity", where all models of a given class are included, with no attempt to weight for either relative skill nor model similarity in the ensemble. Multi-model ensembles tend to outperform single models in overall skill, because weaknesses in any single model are compensated for by other models in the ensemble. However, these ensembles appear to have less diversity than expected (Knutti, 2008).

While intercomparisons of skill scores and climatological patterns are important, these results suggest we need more insight into the nature of similarities and differences between models. The above approaches compare the outputs of the models, but tend to treat the models themselves as black boxes. There are very few representations of the high level designs of global climate models. The Bretherton diagram is perhaps the best known visualization, although it represents an idealized schematic of Earth system processes, rather than the specific design of any model (Cook, 2013). A comparison of the architectural structure of the models should offer useful insights that may help explain observed similarities between model outputs. For example, if the models are architecturally similar, or share significant subcomponents, this would affect the diversity in a multi-model ensemble.

Interest in these architectural patterns is also driven by growing interest in the use of shared infrastructure code in coupled Earth system models (Dickinson et al., 2002). The growing complexity of the coupling task means that couplers require more expertise to develop, and that labs can benefit by comparing their approaches, sharing lessons learnt, and re-using coupling code (Valcke et al., 2012). At the same time, there has been a move towards general reusable subcomponents (e.g. both atmosphere and ocean models using the same numerical solver), compared to earlier model generations, where the code for each component was developed entirely separately. However, code modularity remains a challenge, because nature itself isn't modular. Randall (2011) argues that insufficient attention is given to the challenge of coupling, due to a belief that the science can be contained entirely within modular components.

A discussion of these issues is hampered by a lack of detailed descriptions of the design of Earth system models. While some descriptions of software design are starting to appear (e.g. Drake, 2005), detailed analysis of the design of global climate models remains a challenge because the models have undergone continued code modification for years, and in some cases, decades. This makes a reductionist analysis of specific design decisions impossible, because each design decision is "generatively entrenched" (Lenhard and Winsberg, 2010) – that is, design features form a complex web because each has played a role in generating the others. Furthermore, each lab retains a deep but tacit knowledge base about their own models, which is readily apparent to anyone spending time in that lab (Easterbrook and Johns, 2009), but hard to share through model intercomparison projects. In response to this observation, we argue that a top-down comparative analysis of the architecture of Earth system models is necessary.

3 Methods

For our analysis we selected eight climate models with varying levels of complexity. These include six GCMs which participated in CMIP5 (Taylor et al., 2012), and two EMICs which participated in the EMICAR5 intercomparison project (Eby et al., 2013; Zickfeld et al., 2013). For a summary of information about each model, see Table 1.

We focus on models from the CMIP5 and EMICAR5 ensembles because of the central role these projects play in the IPCC assessment reports.

The first step in analysing each model was preprocessing: stripping out unused code. Each model is a specific configuration of a larger software package (for example, CESM1-BGC is a configuration of the CESM 1.0.5 package) with many available options, including how subgrid-scale eddies are parameterized in the ocean, whether greenhouse gases are updated with emissions or with prescribed concentrations, and whether calculations of ice sheet dynamics are included or excluded. Preprocessing is the first step in the build process, and uses software such as CPP (C Preprocessor) to remove unused options from the code base.

In an earlier version of our analysis (Alexander and Easterbrook, 2011), we used the entire code base for each model, without performing any pre-processing. However, the resulting line counts tended to reflect the number of configuration choices in the code, rather than the size and complexity of the code actually used in a model run. Since we wanted to analyze the models as they were used in the Model Intercomparison Projects, we decided to use preprocessed code for our analysis, to ensure that line count would reflect the size of the actual science code used in the model runs.

The preprocessed code was then analysed using the Understand software (http://www.scitools.com/) in order to extract the dependency structure: which source code files depend on which, through the use of function and subroutine calls. This structure can be interpreted as a directed graph, where any given file calls its "children", is called by its "parents", and has recursively defined "ancestors" and "descendants".

3.1 Classification of source code

In order to sort the code for each model into components (atmosphere, ocean, land, and sea ice), we first identified the top-level driver files for each component. For example, the ocean drivers might consist of a subroutine for initialisation, a subroutine controlling the ocean calculations at each timestep, and a subroutine controlling ocean diagnostics and output. All descendants of the ocean drivers were then classified as the ocean component. Files that were not called by any component, such as the main timestep loop and the flux routines, were classified as the coupler. These top-level files are the drivers of the entire simulation, and control data transfer between components.

Files which were called by multiple components, such as math libraries, parameter lists, and file readers, were classified as shared utilities. Other code often found in shared utilities includes numerical methods used by multiple components. For example, an implicit method commonly used to evaluate advection–diffusion equations (found in both the atmosphere and the ocean) involves solving a tridiagonal matrix. To reduce duplication of similar code, a common practice is to write one tridiagonal matrix solver which is shared by the atmosphere and the ocean.

Within each component, the classification process was repeated to identify subcomponents for atmospheric chemistry, ocean biogeochemistry (BGC), and land vegetation, if these processes were included in the model. Furthermore, sometimes one component was entirely contained in, i.e. controlled by, another: land was frequently treated as a subcomponent of the atmosphere, and sea ice as a subcomponent of the ocean. In the UVic model, sea ice is a subcomponent of the atmosphere; UVic also has a sediment component which is separate from the ocean.

Since any given file might contain several functions and subroutines, circular dependencies between files can and do exist in our analysis. It was necessary to sever some of these dependencies in order for the classification to be reasonable. For example, a low-level file reader might access a function stored in the same file as the top-level program. As a result, the main program file and all of its descendants (i.e., the entire model) would be classified as shared utilities. Only by severing the dependency between the file reader and the main program file could the component structure emerge. The number of dependencies severed was extremely small compared to the total number of dependencies in each model.

3.2 Software diagrams

Using David A. Wheeler's "SLOCCount" tool, we performed a line count (excluding comments and blank lines) of the source code for each component and subcomponent. Then we created diagrams for each model where each component or subcomponent is represented by an ellipse whose area is exactly proportional to the line count of the corresponding code base (See Figs. 1 to 8). The components were assigned standard colours: purple atmosphere, blue ocean, orange land, green sea ice, yellow land ice, red sediment, and grey coupler and shared utilities. Coloured arrows show fluxes between components, which we detected from the coupler code. Note that while each individual diagram is to scale, the diagrams are not to scale with each other. However, each diagram includes a legend below the title which shows the area allocated to one thousand lines of code.

4 Discussion

4.1 Architectural designs

Dividing up a complex system into modules and then arranging these modules hierarchically is an important part of making the world "theoretically intelligible to the human mind" (Simon, 1996). However, there are usually many possible choices of decomposition. While Earth system modellers strive, as Plato suggested, to "carve nature at its joints", in practice, judgment is needed to find a decomposition that is fit for purpose. Comparison of architectural patterns in software has become a standard approach for analyzing the constraints that shape such decisions (Shaw and Garlan, 1996).

The boundaries between components in an Earth system model represent both natural boundaries in the physical world (e.g. the ocean surface), and divisions between communities of expertise (e.g. ocean science vs. atmospheric physics). The model architecture must facilitate simulation of physical processes that cross these boundaries (e.g. heat transport), as well as support collaboration between knowledge communities within the work practices of model development (e.g. to study climate feedbacks). Each major model component tends to have two distinct uses: as a stand-alone component used by a specific subcommunity, and as a building block for coupled Earth system modelling. Hence, there is a tension between the need for each component to remain loosely coupled to facilitate its ongoing use as a stand-alone model, and for tighter integration to study climate interactions with the coupled system.

In our model diagrams (Figs. 1 to 8), two main architectural "shapes" are apparent. First, two of the American GCMs (CESM and GISS; Figs. 1 and 3) have a "star-shaped" architecture: each component is separate from the others, connected only through the central coupler. This design reflects a high level of encapsulation between each component of the model, which is attractive from a software engineering perspective. Once this structure is in place, further changes to any component are relatively easy to incorporate into the coupled model. It also facilitates a mix-and-match approach where, for example, an entirely different ocean model can be substituted with a minimum of effort. In fact, switching between several different ocean components is a common practice at the GISS host institution.

However, a star-shaped architecture can introduce significant challenges when building the coupler: handling fluxes between any combination of four to five components is not a trivial task. These difficulties are alleviated by the "two-sided" architecture present in all three European GCMs (HadGEM, IPSL, and MPI; Figs. 5 to 7). In these models, the only components connected to the coupler are the atmosphere and the ocean; other components are subsets of these two. In all three cases, land is contained within the atmosphere and sea ice is contained within the ocean. When two components share the same grid (spatial discretization), nesting them in this manner is much simpler than routing them through the coupler. This approach also retains the historical paradigm of Atmosphere–Ocean GCMs (AOGCMs) rather than comprehensive Earth System Models (ESMs), even if the model contains all the processes found in an ESM.

The two EMICs (UVic and Loveclim; Figs. 4 and 8) both have intermediate architectures between star-shaped and two-sided. For both models, all components are separate except for sea ice, which is nested within a larger component (atmosphere for UVic, ocean for Loveclim). The atypical structure seen in UVic, where sea ice is treated as a subcomponent of the atmosphere rather than the ocean, was implemented because the sea ice and the atmosphere run on similar time scales. Essentially, UVic nests these components based on their temporal discretization rather than their spatial discretization (which is the same for all components in the model).

4.2 Fluxes

Mass and energy fluxes (represented as arrows, coloured based on their component of origin, in Figs. 1 to 8) are simple in two-sided models: the atmosphere and the ocean both exchange information with the other. The process is more complicated in star-shaped models, because not every component needs to receive data from all of the others. In general, the atmosphere passes fluxes to and from all components with which it shares a boundary (i.e. everything except sediment). The ocean and sea ice are also physically adjacent, so they exchange information in both directions. However, fluxes between the the land and ocean are one-way, since runoff (generally the only land-ocean flux which is represented) moves strictly from the land to the ocean. In GISS (Fig. 3), where a land ice component is also present, it passes runoff either directly to the ocean (including calving) or first to the land.

In GFDL (Fig. 2), quite a different dataflow structure is present. Sea ice is treated as an interface to the ocean: a layer over the entire sea surface which may or may not contain ice. All atmosphere–ocean and land-ocean fluxes must first pass through the sea ice component, even if the fluxes occur at latitudes where sea ice is never actually present. This approach is convenient for interpolation, because the ocean and sea ice components share the same grid, while the atmosphere and land can differ. However, it also uniquely solves the problem of how to represent sea ice – a component immersed in the ocean but with distinct dynamical and physical processes, whose spatial domain is irregular and may change at each timestep.

4.3 Distribution of scientific complexity

Counting lines of code in a given piece of software has been used widely for decades in software engineering. It is used to estimate the amount of effort needed to build software, to measure programmer productivity, and to assess software complexity. Just as often, its validity is questioned, because it is easy to create examples where a program can be improved by making it shorter. However, in practical software development, such examples are unusual. As long as the line counting is done consistently, and comparisons are only made between programs written in the same language and for the same type of application, the number of lines of code can be remarkably useful to assess the size and complexity of a large software system, and to trace its evolution (Park, 1992). Indeed, line count strongly correlates with other measures of software complexity (Herraiz et al., 2007).

These observations allow us to treat line count as a proxy for scientific complexity (the number of physical processes represented in the model). We can see not only a large variation in complexity between models (Fig. 9), but also variations in how complexity is distributed within each model. For all six GCMs, the atmosphere is the largest component. This feature is particularly obvious in HadGEM (Fig. 5), which has a high level of atmospheric complexity due to MetUM's dual use as an operational weather forecasting model. However, both EMICs (UVic and Loveclim; Figs. 4 and 8) have a larger code base for the ocean than for the atmosphere. Since EMICs are built for speed, and atmospheric processes generally require the shortest timesteps in a coupled model, concessions in atmospheric complexity will give the best return on integration time. In other models, particularly CESM (Fig. 1) and IPSL (Fig. 6), the land component is relatively substantial (although not the largest component in the model); this may reflect the growing interest by the modelling community in carbon cycle feedbacks, many of which are terrestrial.

The distribution of complexity among subcomponents can also yield useful insights. Several models, namely CESM (Fig. 1) and HadGEM (Fig. 5), have a substantial code base for atmospheric chemistry. This subcomponent is designed to model processes such as the effects of sulfate aerosol emissions, which are likely to have a large impact on how much warming the planet experiences in the coming decades, but are nonetheless poorly understood (Rosenfeld and Wood, 2013). Other models, including IPSL (Fig. 6) and MPI (Fig. 7), put more weight on the land vegetation and ocean BGC subcomponents. These pieces of code model longer-term processes, such as carbon feedbacks, which are likely to have a large impact on the total amount of warming the planet will experience before it reaches equilibrium (Friedlingstein et al., 2006).

4.4 Shared utilities

The proportion of each model's code base stored as shared utilities also varies widely. On one end of the spectrum, IPSL (Fig. 6) contains no shared utilities at all. The atmosphere and ocean are completely separate components which call no common files. While this approach makes it easy to mix and match components in different configurations of the underlying software package, it also indicates that there is likely some duplication between the atmosphere code and the ocean code, which solve similar fluid dynamics equations.

Conversely, GFDL (Fig. 2) and UVic (Fig. 4) have particularly large proportions of their source code devoted to shared utilities. This is due to the fact that both models contain source code for a custom version of a major utility. GFDL contains a custom MPP (Message Processing Program) to enable parallelization, while UVic contains a custom version of NetCDF, a self-describing data type frequently used for climate model input and output. While most of the other models also use message passing systems and NetCDF libraries, they use unmodified versions which have been preinstalled on the given computing platform. These out-of-the-box utilities are not recompiled for every simulation, and so the source code is not stored with the model. As such, the shared utilities are correspondingly smaller.

4.5 Origin of components

While each coupled model is developed at a home institution (see Table 1), not every component was necessarily developed in-house. It is common practice for one modelling group to adopt another group's ocean component, for example, and modify it to suit the existing coupled architecture. As development continues on the adopted component, the modifications can become substantial, creating a software fork.

Institutions may decide to share components in this manner for several different reasons. Resource constraints, namely a lack of developers to build a new component in-house, particularly affect the smaller modelling groups such as that of UVic (Fig. 4). The UVic ocean component MOM2 (Weaver et al., 2001) is a modified version of a predecessor to GFDL's ocean component MOM4 (Fig. 2), developed in-house by GFDL. UVic also sourced much of its land component (including the vegetation subcomponent TRIFFID) from code written at the Hadley Centre (Meissner et al., 2003), much of which is present in HadGEM (Fig. 5). However, large modelling groups adopt components from other institutions as well. The CESM ocean POP2 (Smith et al., 2010) and sea ice CICE4 (Bailey et al., 2010) components were both built at Los Alamos National Laboratory, rather than the National Centre for Atmospheric Research (CESM's host institution), and reflect NCAR's goal of creating and supporting a community model.

In recent years, there have also been organized collaborations between institutions to build shared components with high levels of scientific complexity. These components are then included in several coupled modelling systems, and typically can also be run in stand-alone configurations. For example, IPSL (Fig. 6) and MPI (Fig. 7) both use the OASIS coupler (Valcke, 2013), developed by scientists from the French institutions CERFACS and CNRS. IPSL also uses NEMO (Madec, 2008; Vancoppenolle et al., 2008), an ocean and sea ice model developed by a consortium of five European institutions. HadGEM (Fig. 5), which consists almost entirely of in-house components in this configuration (the UKCA atmospheric chemistry subcomponent is the only major piece of code developed externally), is currently incorporating OASIS, NEMO, and CICE into its next release (Hewitt et al., 2011).

5 Conclusions

These software architecture diagrams show, in a broad sense, how climate models work: how the climate system is divided into components and how these components communicate with each other. They also illustrate the similarities and differences between the eight models we have analysed. Some models, particularly in North America, exhibit a high level of encapsulation for each component, with all communication managed by the coupler. Other models, particularly in Europe, implement a binary atmosphere–ocean architecture which simplifies the coupling process. Institutions focus their efforts on different climatic processes, which eventually causes different components and subcomponents to dominate each model's source code. However, not all models are completely independent of each other: modelling groups commonly exchange pieces of code, from individual routines up to entire components. Finally, climate models vary widely in complexity, with the total line count varying by a factor of 20 among the eight models we analysed.

Our analysis also offers some new insights into the question of model diversity, which is important when creating multi-model ensembles. Masson and Knutti (2011) showed that models from the same lab tend to have similar climatology, even over multiple model generations. We believe this can be explained in terms of their architectural structure and, in particular, the distribution of complexity within the model. We hypothesize that the relative size of each component within an Earth system model indicates the relative size of the pool of expertise available to that lab in each Earth system domain (once adjustments are made for components imported from other labs). The availability of different areas of expertise at each lab would provide a sufficient explanation for the clustering effects reported by Masson and Knutti (2011).

Our diagrams may prove to be useful for public communication and outreach by their host institutions. The inner workings of climate models are rarely discussed in the media, even by science reporters; as such, these pieces of software are fundamentally mysterious to most members of the public. Additionally, the diagrams could be used for communication between scientists, both within and across institutions. It can be extremely useful for climate scientists, whether they are users or developers of coupled models, to understand how other modelling groups have addressed the same scientific problems. A better understanding of the Earth system models used by other institutions may open doors for international collaboration in the years to come.

Appendix A: Accessing model code

The procedure for obtaining climate model source code varies between institutions. Below are instructions for requesting access to each model.

CESM: Complete the registration form at http://www.cesm.ucar.edu/models/register/register_cesm.cgi. Instructions for accessing NCAR's Subversion repository will then be provided via email.

GFDL: Source code is open access through a GitHub repository; instructions are available at http://www.mom-ocean.org/web/downloads.

GISS: The AR5 branch of ModelE (the software package underlying GISS-E2-R-TCADI) can be downloaded as a compressed file from http://simplex.giss.nasa.gov/snapshots/.

UVic: A compressed file containing the source code can be downloaded via a password-protected link at http://climate.uvic.ca/model/. This page contains instructions for requesting a password via email.

HadGEM: Obtaining source code for climate models developed at the UK Met Office requires signing a user agreement. Contact Tim Johns for more information.

IPSL: Installation scripts can be downloaded through IPSL's Subversion repository, as described at http://forge.ipsl.jussieu.fr/igcmg/wiki/platform/en/documentation/C_installation. In order for these scripts to fully extract the model source code, a username and password must be requested via email.

MPI: Obtaining source code for climate models developed at the Max Planck Institut für Meteorologie requires signing a user agreement. Contact Reinhard Budich for more information.

Loveclim: Contact Pierre-Yves Barriat at the Université catholique de Louvain to request access to the Loveclim source code.

Acknowledgements. We are grateful to the model developers who granted us access to source code, assisted with preprocessing, answered questions about their models, and provided feedback on the diagrams: Mariana Vertenstein and Jim Edwards from the National Centre for Atmospheric Research; Venkatramani Balaji from the Geophysical Fluid Dynamics Laboratory; Gavin Schmidt from the NASA Goddard Institute for Space Studies; Michael Eby from the University of Victoria; Tim Johns from the UK Met Office; Arnaud Caubel, Marie-Alice Foujols, and Anne Cozic from the Institut Pierre Simon Laplace; Reinhard Budich and Marco Giorgetta from the Max Planck Institut für Meteorologie; and Pierre-Yves Barriat from the Université catholique de Louvain. This project was supported by the Centre for Global Change Science at the University of Toronto as well as the National Sciences and Engineering Research Council of Canada.

References

Alexander, K. and Easterbrook, S. M.: The Software Architecture of Global Climate Models, in: AGU Fall Meeting 2011, San Francisco, USA, Abstract ID 1204770, 2011. 356

Bailey, D., Holland, M., Hunke, E., Lipscomb, B., Briegleb, B., Bitz, C., and Schramm, J.: Community Ice CodE (CICE) User's Guide Version 4.0, Tech. Rep., National Center for Atmospheric Research, available at: http://www.cesm.ucar.edu/models/cesm1.0/cice/ice_usrdoc.pdf (last access: 19 November 2014), 2010. 363

Collins, M.: Ensembles and probabilities: a new era in the prediction of climate change, Philos. T. R. Soc. A, 365, 1957–70, doi:10.1098/rsta.2007.2068, 2007. 354

Cook, K. H.: An introduction to the climate system 1, in: Climate Dynamics, Princeton University Press, 1–3, 2013. 354

Dickinson, R. E., Zebiak, S., Anderson, J., Blackmon, M., De Luca, C., Hogan, T., Iredell, M., Ji, M., Rood, R., Suarez, M., and Taylor, K. E.: How can we advance our weather and climate models as a community?, B. Am. Meteorol. Soc., 83, 431–434, doi:10.1175/1520-0477(2002)083<0431:HCWAOW>2.3.CO;2, 2002. 354

Drake, J. B.: Overview of the software design of the community climate system model, Int. J. High Perform. C., 19, 177–186, doi:10.1177/1094342005056094, 2005. 355

Easterbrook, S. M. and Johns, T. C.: Engineering the software for understanding climate change, Comput. Sci. Eng., 11, 65–74, doi:10.1109/MCSE.2009.193, 2009. 355

Eby, M., Weaver, A. J., Alexander, K., Zickfeld, K., Abe-Ouchi, A., Cimatoribus, A. A., Crespin, E., Drijfhout, S. S., Edwards, N. R., Eliseev, A. V., Feulner, G., Fichefet, T., Forest, C. E., Goosse, H., Holden, P. B., Joos, F., Kawamiya, M., Kicklighter, D., Kienert, H., Matsumoto, K., Mokhov, I. I., Monier, E., Olsen, S. M., Pedersen, J. O. P., Perrette, M., Philippon-Berthier, G., Ridgwell, A., Schlosser, A., Schneider von Deimling, T., Shaffer, G., Smith, R. S., Spahni, R., Sokolov, A. P., Steinacher, M., Tachiiri, K., Tokos, K., Yoshimori, M., Zeng, N., and Zhao, F.: Historical and idealized climate model experiments: an intercomparison of Earth system models of intermediate complexity, Clim. Past, 9, 1111–1140, doi:10.5194/cp-9-1111-2013, 2013. 352, 353, 355

Friedlingstein, P., Cox, P., Betts, R., Bopp, L., von Bloh, W., Brovkin, V., Cadule, P., Doney, S., Eby, M., Fung, I., Bala, G., John, J., Jones, C., Joos, F., Kato, T., Kawamiya, M., Knorr, W., Lindsay, K., Matthews, H. D., Raddatz, T., Rayner, P., Reick, C., Roeckner, E., Schnitzler, K.-G., Schnur, R., Strassmann, K., Weaver, A. J., Yoshikawa, C., and Zeng, N.: climate–carbon cycle feedback analysis: results from the C^{4}MIP model intercomparison, J. Climate, 19, 3337–3353, doi:10.1175/JCLI3800.1, 2006. 362

Herraiz, I., Gonzalez-Barahona, J. M., and Robles, G.: Towards a theoretical model for software growth, in: Proceedings of the Fourth International Workshop on Mining Software Repositories, MSR '07, IEEE Computer Society, Washington, DC, USA, doi:10.1109/MSR.2007.31, 2007. 361

Hewitt, H. T., Copsey, D., Culverwell, I. D., Harris, C. M., Hill, R. S. R., Keen, A. B., McLaren, A. J., and Hunke, E. C.: Design and implementation of the infrastructure of HadGEM3: the next-generation Met Office climate modelling system, Geosci. Model Dev., 4, 223–253, doi:10.5194/gmd-4-223-2011, 2011. 364

Knutti, R.: Should we believe model predictions of future climate change?, Philos. T. R. Soc. A, 366, 4647–64, doi:10.1098/rsta.2008.0169, 2008. 354

Lenhard, J. and Winsberg, E.: Holism, entrenchment, and the future of climate model pluralism, Stud. Hist. Philos. M. P., 41, 253–262, doi:10.1016/j.shpsb.2010.07.001, 2010. 355

Madec, G.: NEMO Ocean Engine, Tech. rep., Laboratoire d'Oceanographie et du Climat: Experimentation et Approches Numeriques, 2008. 363

Masson, D. and Knutti, R.: Climate model genealogy, Geophys. Res. Lett., 38, 1–4, doi:10.1029/2011GL046864, 2011. 354, 364

Meissner, K. J., Weaver, A. J., Matthews, H. D., and Cox, P. M.: The role of land surface dynamics in glacial inception: a study with the UVic Earth System Model, Clim. Dynam., 21, 515–537, doi:10.1007/s00382-003-0352-2, 2003. 363

Park, R.: Software Size Measurement: a Framework for Counting Source Statements, Tech. Rep. CMU/SEI-92-TR-020, Software Engineering Institute, Carnegie Mellon University, available at: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11689 (last access: 19 November 2014), 1992. 361

Randall, D. A.: Should climate models be open source?, IEEE Software, 28, 62–65, doi:10.1109/MS.2011.144, 2011. 355

Reichler, T. and Kim, J.: How well do coupled models simulate today's climate?, B. Am. Meteorol. Soc., 89, 303–311, doi:10.1175/BAMS-89-3-303, 2008. 353

Rosenfeld, D. and Wood, R.: Aerosol cloud-mediated radiative forcing: highly uncertain and opposite effects from shallow and deep clouds, in: Climate Science for Serving Society, edited by: Asrar, G. R., and Hurrell, J. W., Springer Netherlands, Dordrecht, doi:10.1007/978-94-007-6692-1, 2013. 362

Shaw, M. and Garlan, D.: Software architecture: perspectives on an emerging discipline, Prentice Hall, 1996. 358

Simon, H. A.: The Sciences of the Artificial, MIT Press, 1996. 358

Smith, R., Jones, P., Briegleb, B., Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden, C., Fox-Kemper, B., Gent, P., Hecht, M., Jayne, S., Jochum, M., Large, W., Lindsay, K., Maltrud, M., Norton, N., Peacock, S., Vertenstein, M., and Yeager, S.: The Parallel Ocean Program (POP) Reference Manual: Ocean Component of the Community Climate System Model (CCSM) and Community Earth System Model (CESM), Tech. rep., Los Alamos National Laboratory, Report Number LAUR-10-01853, available at: http://www.cesm.ucar.edu/

Title Page
Abstract Introduction
Conclusions References
Tables Figures
◂◂ ▸▸
◂ ▸
Back Close
Full Screen / Esc
Printer-friendly Version
Interactive Discussion

models/ccsm4.0/pop/doc/sci/POPRefManual.pdf (last access: 19 November 2014), 2010. 363

Taylor, K. E., Stouffer, R. J., and Meehl, G. A.: An overview of CMIP5 and the experiment design, B. Am. Meteorol. Soc., 93, 485–498, doi:10.1175/BAMS-D-11-00094.1, 2012. 352, 353, 355

Valcke, S.: The OASIS3 coupler: a European climate modelling community software, Geosci. Model Dev., 6, 373–388, doi:10.5194/gmd-6-373-2013, 2013. 363

Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O'Kuinghttons, R., Riley, G. D., and Vertenstein, M.: Coupling technologies for Earth System Modelling, Geosci. Model Dev., 5, 1589–1596, doi:10.5194/gmd-5-1589-2012, 2012. 355

Vancoppenolle, M., Fichefet, T., Goosse, H., Bouillon, S., Beatty, C. K., and Maqueda, M. A. M.: LIM3, an advanced sea-ice model for climate simulation and operational oceanography, Mercator Quarterly Newsletter, 28, 16–21, 2008. 363

Weaver, A. J., Eby, M., Wiebe, E. C., Bitz, C. M., Duffy, P. B., Ewen, T. L., Fanning, A. F., Holland, M. M., MacFadyen, A., Matthews, H. D., Meissner, K. J., Saenko, O., Schmittner, A., Wang, H., and Yoshimori, M.: The UVic earth system climate model: model description, climatology, and applications to past, present and future climates, Atmos. Ocean, 39, 361–428, doi:10.1080/07055900.2001.9649686, 2001. 363

World Climate Research Program: WCRP Working Group on Coupled Modeling Catalogue of Model Intercomparison Projects (MIPs), available at: http://www.wcrp-climate.org/wgcm/projects.shtml (last access: 19 November 2014), 2014. 353

Zickfeld, K., Eby, M., Weaver, A. J., Alexander, K., Crespin, E., Edwards, N. R., Eliseev, A. V., Feulner, G., Fichefet, T., Forest, C. E., Friedlingstein, P., Goosse, H., Holden, P. B., Joos, F., Kawamiya, M., Kicklighter, D., Kienert, H., Matsumoto, K., Mokhov, I. I., Monier, E., Olsen, S. M., Pedersen, J. O. P., Perrette, M., Philippon-Berthier, G., Ridgwell, A., Schlosser, A., Schneider Von Deimling, T., Shaffer, G., Sokolov, A., Spahni, R., Steinacher, M., Tachiiri, K., Tokos, K. S., Yoshimori, M., Zeng, N., and Zhao, F.: Long-term climate change commitment and reversibility: an EMIC intercomparison, J. Climate, 26, 5782–5809, doi:10.1175/JCLI-D-12-00584.1, 2013. 352, 353, 355

Title Page
Abstract | Introduction
Conclusions | References
Tables | Figures
◂◂ | ▸▸
◂ | ▸
Back | Close
Full Screen / Esc
Printer-friendly Version
Interactive Discussion

Table 1. Information about each model including its full configuration name and an abbreviation for use in the text, level of complexity (GCM or EMIC), and the name and location of its host institution.

Model Name (Abbreviation)	Complexity	Institution	Country
CESM1-BGC (CESM)	GCM	National Centre for Atmospheric Research	USA
GFDL-ESM2M (GFDL)	GCM	Geophysical Fluid Dynamics Laboratory	USA
GISS-E2-R-TCADI (GISS)	GCM	NASA Goddard Institute for Space Studies	USA
UVic ESCM 2.9 (UVic)	EMIC	University of Victoria	Canada
HadGEM2-ES (HadGEM)	GCM	Hadley Centre for Climate Prediction and Research	UK
IPSL-CM5A-LR (IPSL)	GCM	Institut Pierre Simon Laplace	France
MPI-ESM-LR (MPI)	GCM	Max Planck Institut für Meteorologie	Germany
Loveclim 1.3 (Loveclim)	EMIC	Université catholique de Louvain	Belgium

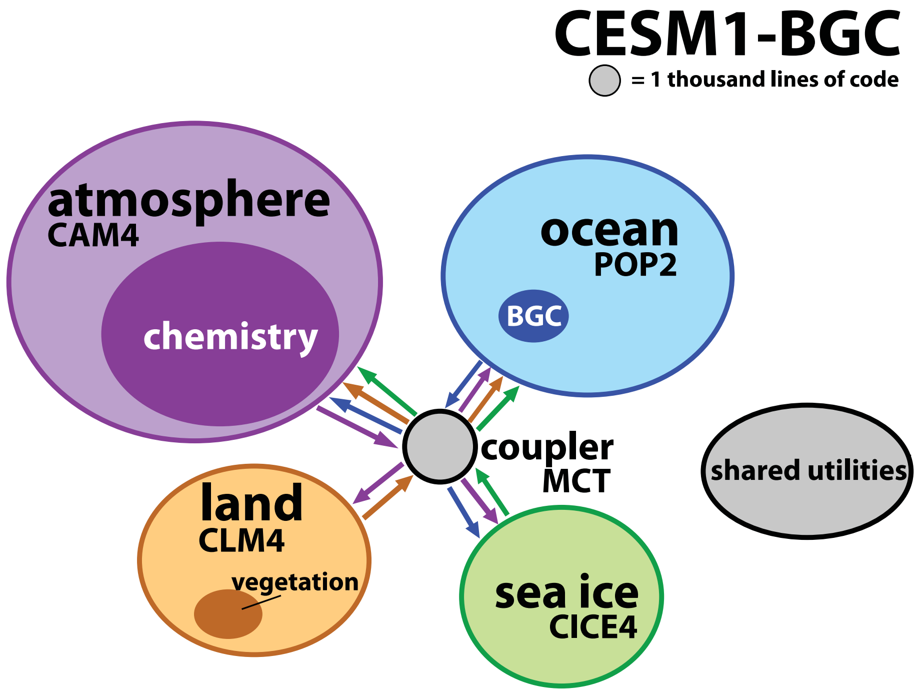

Figure 1. Architecture diagram for CESM1-BGC.

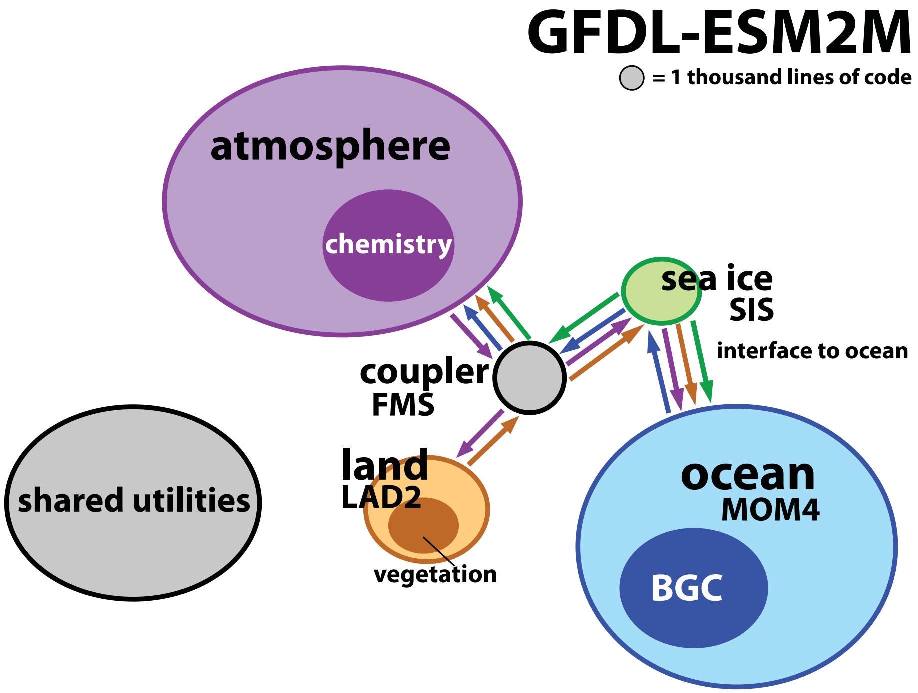

Figure 2. Architecture diagram for GFDL-ESM2M.

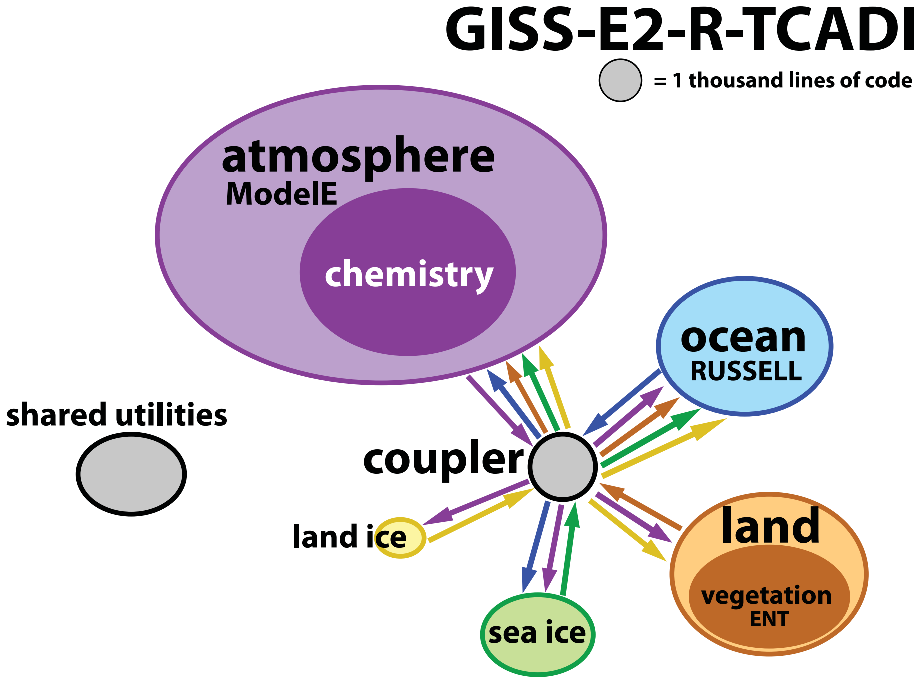

Figure 3. Architecture diagram for GISS-E2-R-TCADI.

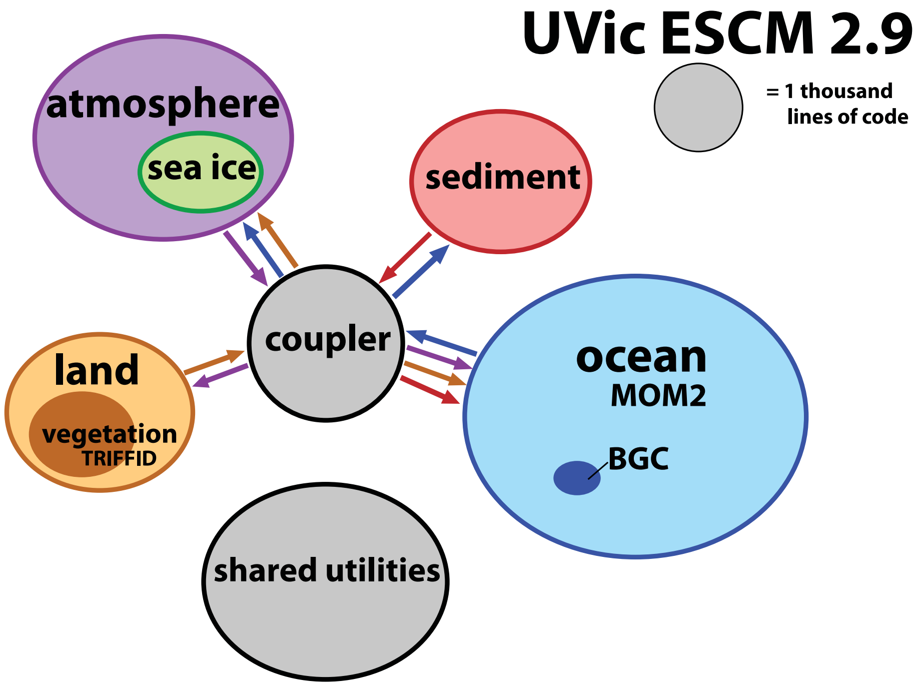

Figure 4. Architecture diagram for UVic ESCM 2.9.

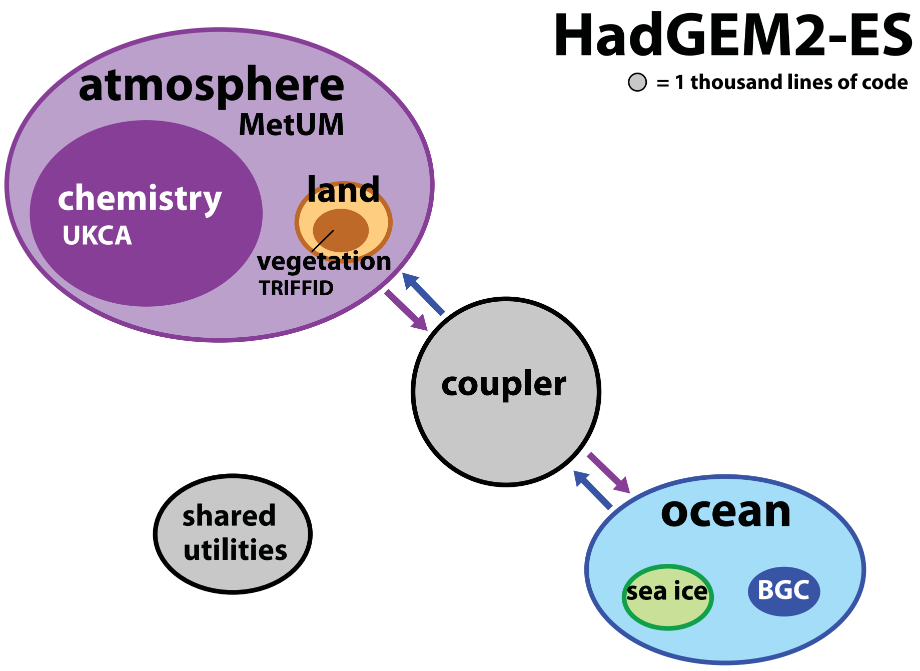

Figure 5. Architecture diagram for HadGEM2-ES.

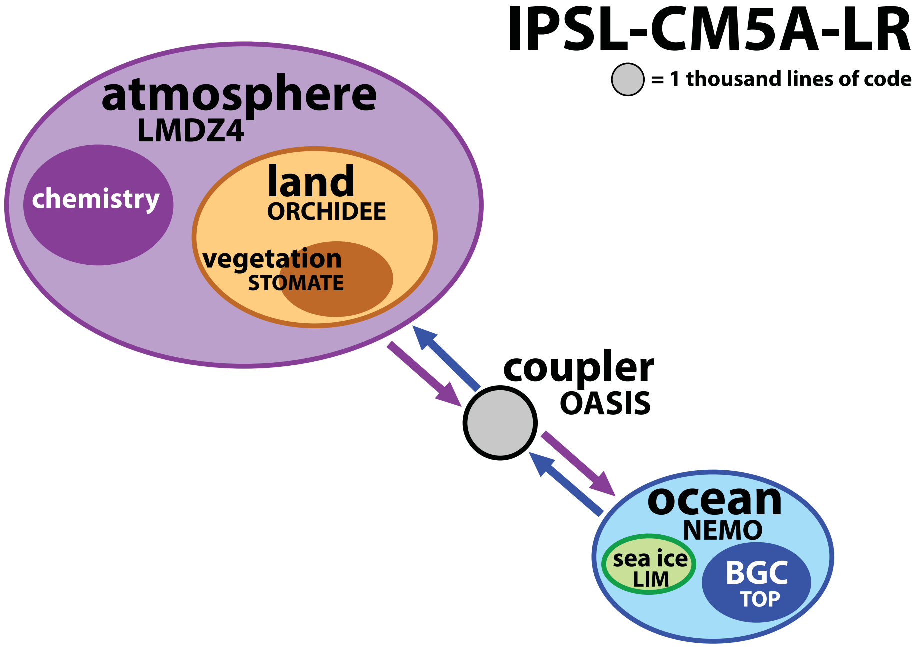

Figure 6. Architecture diagram for IPSL-CM5A-LR.

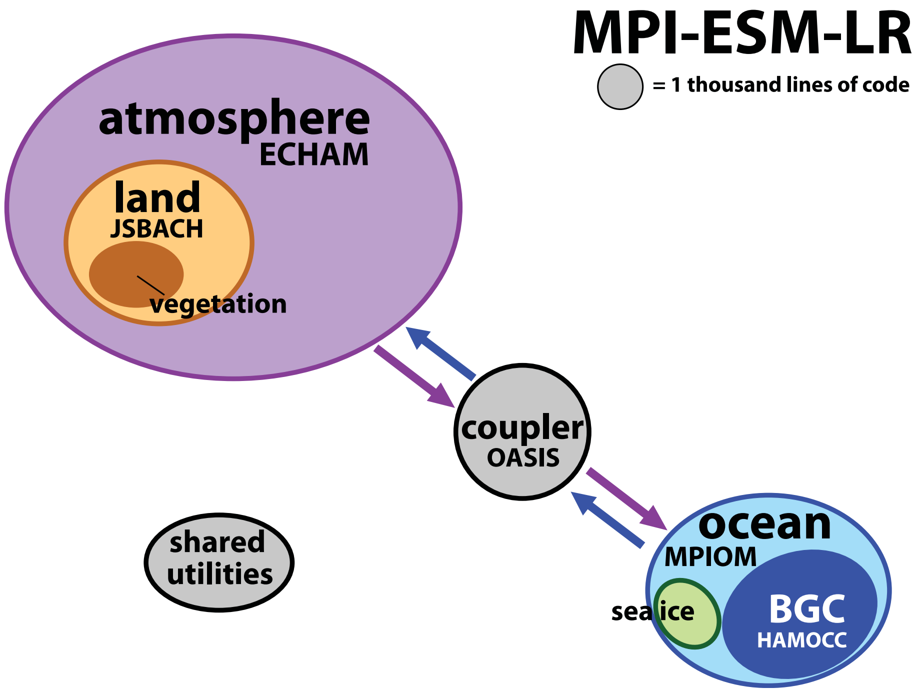

Figure 7. Architecture diagram for MPI-ESM-LR.

GMDD
8, 351–379, 2015

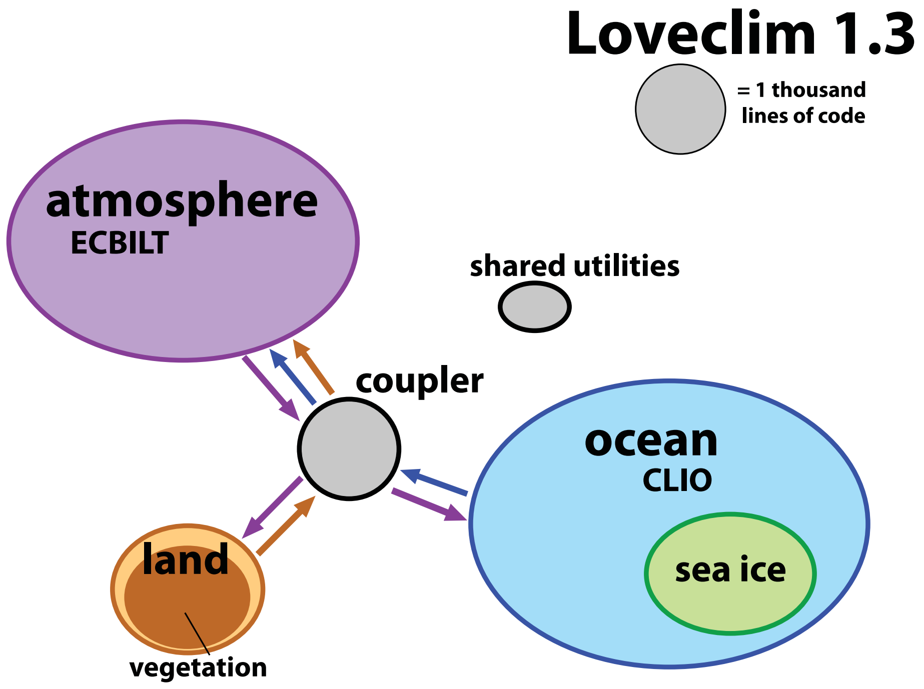

Figure 8. Architecture diagram for Loveclim 1.3.

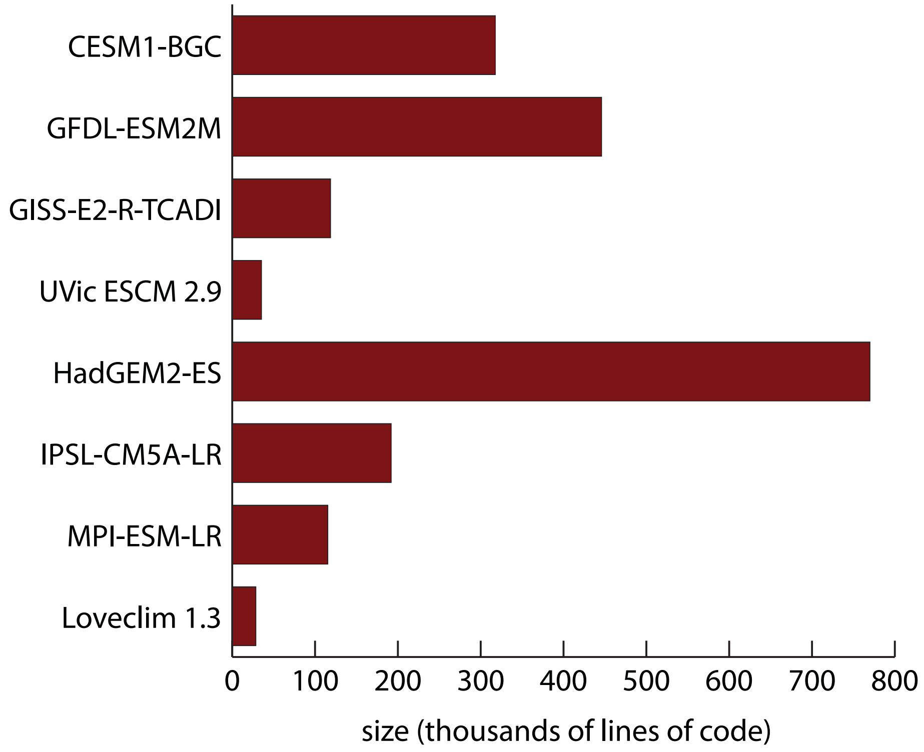

Figure 9. Line count of the source code of each model, excluding comments and blank lines. Generated using David A. Wheeler's "SLOCCount".