CIME (Common Infrastructure for Modeling the Earth): What is it?

Mariana Vertenstein
CESM Software Engineering Group (CSEG)

CIME is an emerging paradigm for the collective construction and maintenance of the software infrastructure required for earth system model development and application.

CIME was developed as NCAR/CSEG’s response to the appeal from the February 2015 Global Modeling summit of the US Global Change Research Program (USGCRP) / Interagency Group on Integrative Modeling (IGIM) for greater coordination among centers engaged in Earth system modeling.

CIME enables collaborations on the development and open distribution of infrastructure (where there is generally no intellectual property) by separating it from prognostic component model code and all of its scientific developments with intellectual property.

CIME facilitates communication and collaboration on the design and implementation of earth system model infrastructure software.

CIME reduces duplication and redundancy in the development and maintenance of infrastructure associated with earth system models.

CIME code is publicly readable but has restricted write access.

CIME governance will be established collectively to satisfy the requirements of its adherents and their sponsors, as well as the vision of USGCRP/IGIM. Meanwhile, the head of CSEG will provide pro tem governance.

As detailed below CIME is composed of:

1. Flexible hub and spoke inter-component coupling architecture
2. Data components that permit flexible instantiation of coupling feedbacks
3. Scripting, system test, and unit test infrastructure
4. HPC and other utilities (e.g., regridding, verification, load-balancing)
5. Stand-alone development and testing platform

CIME does not contain

- unpublished intellectual property
- prognostic earth-system model components
- intra-model coupling
- direct component to component coupling
- hierarchical coupling capabilities
The CIME Paradigm

1. CIME Coupling Architecture

CIME contains inter-component coupling capability through a hub-and-spoke architecture for models run as a single executable. In this architecture, the components are the spokes and the central hub is a coupler (or mediator in ESMF/NUOPC parlance). Implicit in this architecture is a top-level driver that controls the temporal evolution of the model system and that runs on the union of all component processors. Components can be run sequentially, concurrently, or in some mixed sequential/concurrent layout on a set of processors or on unique grids. The coupler/mediator can be run either on all or a subset of the total processors and performs mapping between components, merging, flux calculations, and diagnostics. Currently, CIME provides the ability to couple seven component models (atmosphere, land, river, land-ice, sea-ice, ocean, surface wave).

In general, there are three options for each component model; a prognostic model external to CIME; an internal “data model” that reads in observational or model generated data; or an internal “stub model” that simply satisfies interface requirements. The CIME coupling architecture is independent of the option chosen for each component, which allows for the flexible control of feedbacks between prognostic components. Furthermore, CIME can be run stand-alone by specifying all components as either data or stub models.

CIME does not allow direct component-to-component communication. It also does not yet contain intra-component coupling infrastructure. For example, the coupling between physics and dynamics in the Community Atmosphere Model (CAM) is CAM specific and not part of CIME. However, moving forward, CIME is targeting adoption of new hierarchical coupling functionality, where components themselves contain a top-level driver that is a “child” of the top-level CIME “parent” driver. These child drivers become the component “spokes” for the CIME driver. As an example, the CAM chemistry could be refactored to run as a sub-component of CAM on potentially disjoint processor sets from the rest of the CAM code and a CAM “child” driver would be created to control the temporal evolution and communication of the CAM chemistry and the rest of CAM. The CIME driver would call the CAM driver, which in turn would call the CAM components. This hierarchical coupling capability would become the standard for intra-component coupling. We anticipate using NUOPC/ESMF as the basis for this capability.

2. CIME data and stub components

As mentioned above, the ability to selectively turn two-way feedbacks on and off between prognostic components is achieved via the replacement of one or more components with either data or stub components. Although CIME is NOT accompanied by any prognostic components, it does contain the full suite of data and stub model components that can be flexibly utilized by the CIME infrastructure to define and control feedbacks. More importantly, this flexible design permits feedbacks between components to be selectively turned off, thereby enabling
researchers to address a broad range of scientific questions and to meet specific requirements.

3. CIME Scripting and Testing Infrastructure

CIME contains a scripting infrastructure that enables users to easily create, build and run numerous out-of-the-box experiments encompassing a wide range of model resolutions, component configurations and model feedbacks. In addition, CIME provides users with a standardized mechanism for custom configuration of individual experiments along with options for both short and long term archiving of model data. Such capability is key for community usage of complex earth system models.

CIME is also accompanied by very useful automated system and unit testing frameworks that allow users to run whole test suites with one command. For any complex software system, and especially for a coupled modeling system, automated system and unit tests to verify infrastructure functionality on a variety of machines and compilers are essential. These tests satisfy a number of high-level software requirements, such as the ability to give bit-for-bit answers after restarting from a checkpoint, providing answers independent of processor and/or OpenMP thread count, and ensuring that code refactorings do not change answers unless they are expected to do so.

4. CIME Utilities

CIME is currently accompanied by a growing collection of important infrastructure utilities that need to be maintained. A few key examples are:

- **PIO**: Since limiting external storage access to a single master processor creates a serial bottleneck, an efficient parallel I/O subsystem is a critical component of parallel earth system models. CIME contains a parallel I/O library called PIO, which is the main I/O library in both CESM and ACME components (including MPAS).
- **ECT**: A new ensemble-based consistency test (CESM Ensemble Consistency Test, ECT) provides a critical new capability to formalize and simplify the software verification process of climate models. ECT is a new objective tool for the evaluation of climate consistency in porting and model development work where bit-for-bit answers are not expected. It is, therefore, a replacement for the historical approach that was both subjective and computationally expensive.
- **Load Balancing**: Determining the optimal number of CPU cores to allocate to earth system model components (i.e. achieving an even load-balance) can have a dramatic impact on the computational cost of a model simulation. A static load-balancing tool provides an effective means of boosting scalability and performance of the code and is incorporated into CIME.
- **Regridding**: The current CIME coupling infrastructure provides parallel runtime regridding capabilities, but utilizes mapping files that are generated offline. Leveraging ESMF offline parallel regridding utilities, CIME provides a simple tool chain for the user to rapidly generate all necessary mapping files needed for a new model configuration.
5. Stand-alone capability

CIME can be downloaded, tested and developed in a stand-alone mode, without any dependence on prognostic model components. In this mode, the coupling infrastructure only utilizes the CIME data and/or stub model components. This capability has proven vital in facilitating collaboration on CIME with non-CESM modeling efforts (such as DOE/ACME and possibly NOAA/NEMS).

The Current State of CIME

CIME is contained in a publicly viewable and downloadable github repository. This is a key feature since it permits all infrastructure developments and issues to be transparent. All improvements, bug fixes and new options, are tracked and communicated in the github repository via github’s wiki and issue tracker system. Users can easily choose to subscribe and receive automatic notifications of any of these updates.

Although CIME is now world readable, write access is restricted. However, there are mechanisms in place permitting modeling efforts to contribute back to CIME. For example, to actively collaborate, developers would simply issue a pull request on github. This procedure is documented at https://github.com/CESM-Development/cime/wiki/CIME-Development-Guide.

Anyone may issue a pull request. To date, the gatekeeper for requests is the pro tem governance. However, this governance can be extended depending on the collaboration. As an example, CESM and the DOE/ACME project have created a new github repository, https://github.com/ESMCI/cime, that mirrors CIME, but where the determination of what pull requests to accept are made jointly by both CSEG and ACME software engineers. A similar CIME repository structure is also in the process of being established with NOAA. The CIME administrative structure can be easily enhanced using github’s administration and organizational tools.

Although CSEG has established the first prototype workflow for using and developing CIME (https://github.com/CESM-Development/cime/wiki/CIME-Git-Workflow), we expect that as more collaborators become involved in development, this workflow will evolve to accommodate new requirements.

It is important to stress that CIME is in its infancy. As it continues to expand in the number and variety of potential users and contributors, a formal governance structure will need to be established by all stakeholders.