The Software Architecture of Global Climate Models
Kaitlin Alexander, Steve Easterbrook
Software Engineering Lab, Department of Computer Science, University of Toronto

1. What is a Climate Model?

We don’t have access to multiple Earths for the purpose of experimentation. Instead, scientists have developed global climate models (GCMs): large pieces of software that simulate the climate system, and how it might react to agents of climate change. In this study, we compared and contrasted the software architecture of seven GCMs from Canada, the United States, and Europe.

Common Features

- **Infrastructure code** (shell scripts and Perl) configures, builds, and runs the model. **Scientific code** (FORTRAN) is present in the top-level program for the main loop of the simulation. Components often use different grids or resolutions, so direct interaction is difficult.

- **Cells** (3D, ~100 km wide) are created by laying a grid over the Earth’s surface and atmosphere. **Time steps** indicate how often calculations are performed (typically minutes to hours).

- **Dynamics** calculations resolve fluid dynamics from first principles. **Physics** calculations are parameterizations: approximations for complex or small-scale processes.

- **Couplers** manage component interaction and fluxes builds, and runs the model. Each iteration, it calls components one by one, instructing them to complete their calculations for one time step and update fluxes.

- **Radiative forcings** are passed to components with plain arrows.

Component-Based Software Engineering (CBSE)

A climate model is really a collection of models (components) for the atmosphere, land, etc. They are highly encapsulated, for stand-alone use as well as a mix-and-match approach that facilitates code sharing. CBSE pools resources, creating high-quality components that are used by many GCMs. Components are modified when they are passed between institutions, to suit new GCMs. These modifications are encouraged by code sharing practices. Virtually anyone can get access to GCM source code, but only the core development team can modify the master copy.

A drawback of CBSE is the fact that the real world is not encapsulated. Relationships between sea ice and the ocean are particularly difficult to represent. Here are some examples of the different approaches taken:

- **CESM**: Sea ice is separate to the ocean, with a transient boundary.
- **IPSL**: Sea ice is a sub-component of the ocean.
- **GFDL**: Sea ice is an interface to the ocean. All fluxes to and from the ocean pass through “sea ice”, even if no ice is actually present.

Conclusions

While some features of software architecture are common among every GCM we studied, other features show a wide range of different design choices. Coupler structure, distribution of complexity, and levels of component encapsulation all vary widely.

These architectural differences may provide new insights into variability and spread between model results. By examining software variations, as well as scientific variations, we can better understand discrepancies in GCM output.