Best Practices in the Community Earth System Model

Nancy Norton
CESM Software Engineering Group (CSEG)
Climate and Global Dynamics (CGD)
NCAR Earth Systems Laboratory (NESL)
National Center for Atmospheric Research (NCAR)

The National Center for Atmospheric Research is sponsored by the National Science Foundation

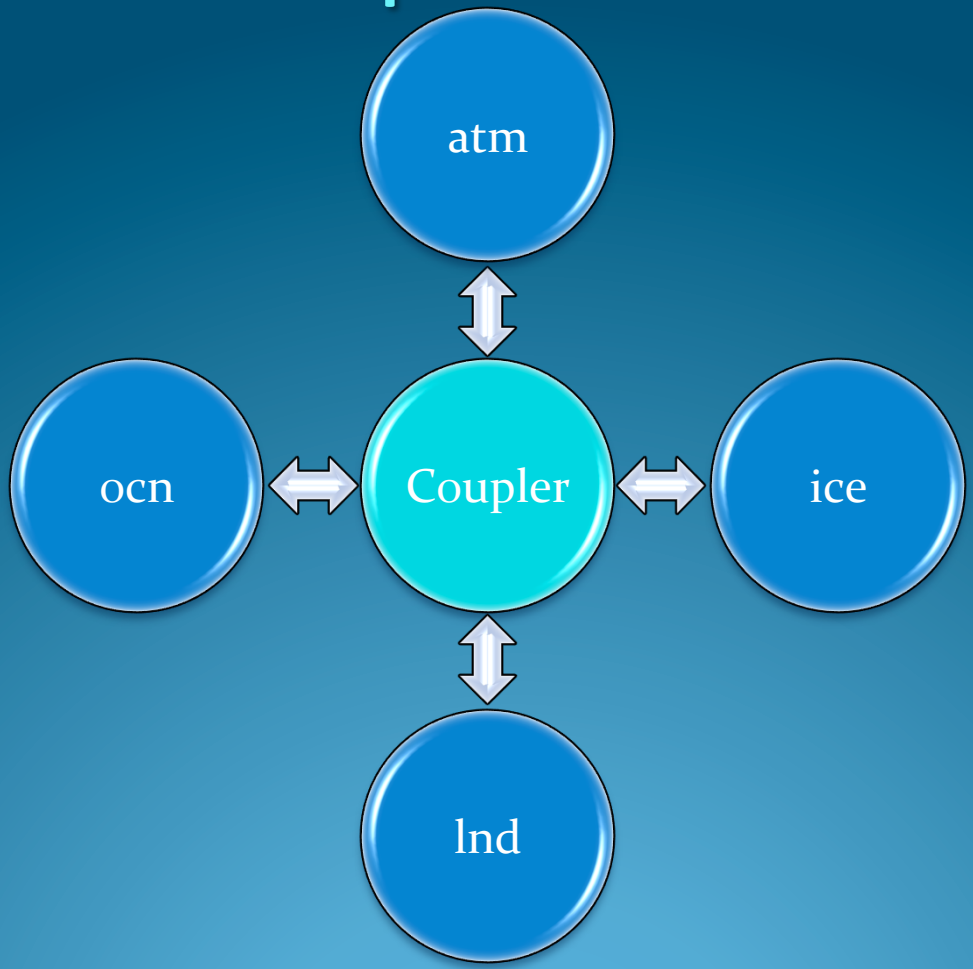
First Things First

- Diversity of software engineering in UCAR
- Introduction to CESM
  - What is CESM?
  - What/who is CSEG?
  - What does CSEG do?
- Complexity
- Best Practices
- Advice, Discussion, Improvements

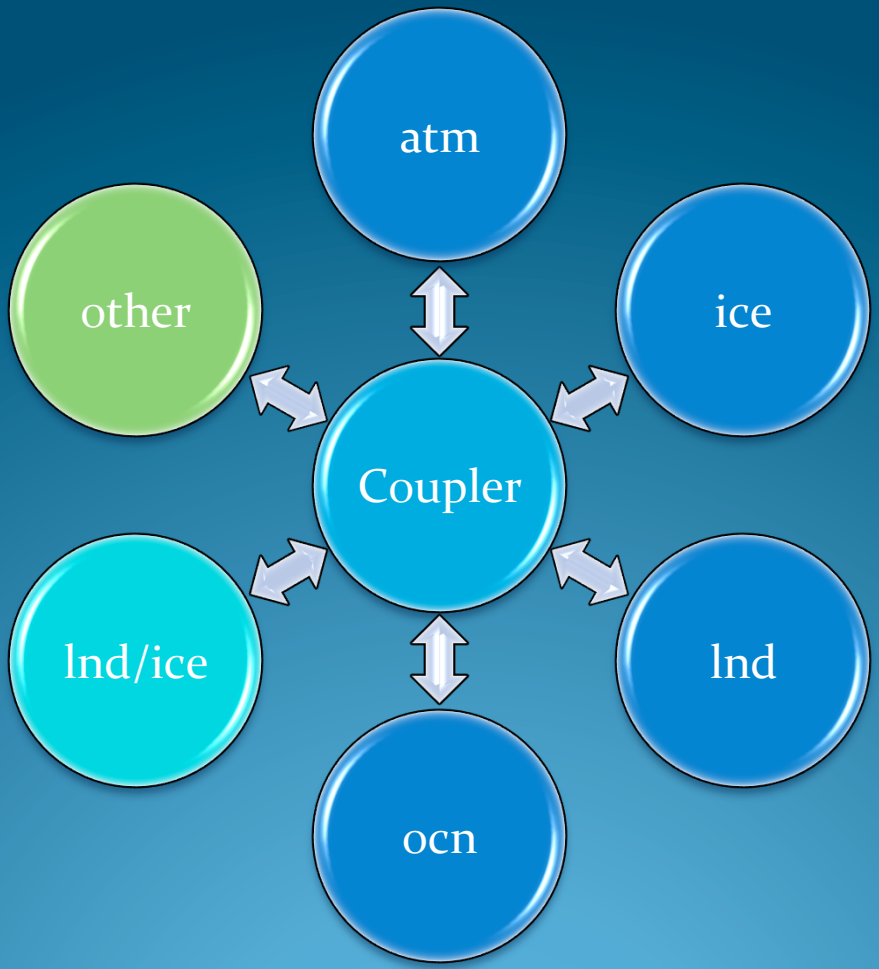
What is CESM?

The Community Earth System Model (CESM) is a fully-coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states.

Simplest Conceptual Model of CESM



Conceptual Model of CESM



History of CESM

- CSM: **Climate System Model**
  - CSM1.0 June 1996
  - CSM1.2 July 1998
  - CSM1.4 July 2000
- CCSM: **Community Climate System Model**
  - CCSM2.0 May 2002
  - CCSM2.0.1 October 2002
  - CCSM3.0 June 2004
  - CCSM4.0 April 2010
- CESM: **Community Earth System Model**
  - CESM1.0 June 2010
  - CESM1.0.1 September 2010
  - CESM1.0.2 Dec 2010
  - CESM1.0.3 June 2011
- **Coming Soon:**
  - CESM1.0.4 2011
  - CESM 1.1 2012

CESM Development Cycle

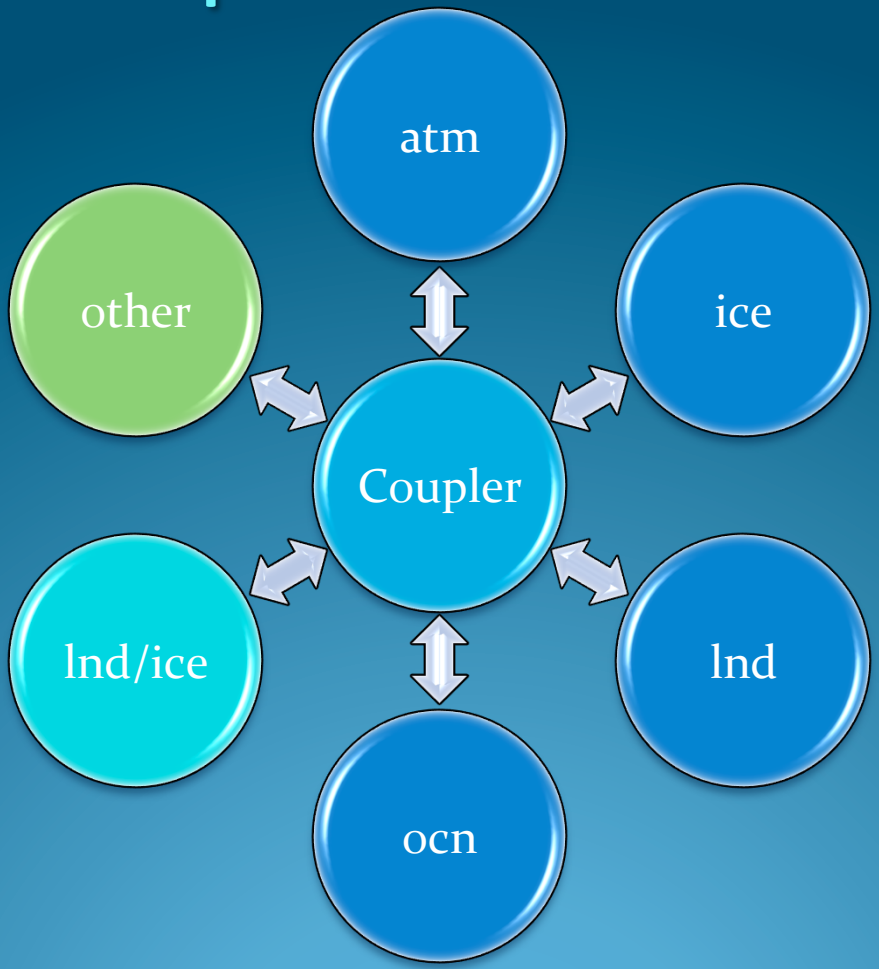
Simplified CESM Development Cycle:

- Model development (new physics, numerics) phase
- Climate simulations ("production runs")
- Analysis/validation (IPCC, scientific papers and presentations)
- Public release of model
- Public release of model output data

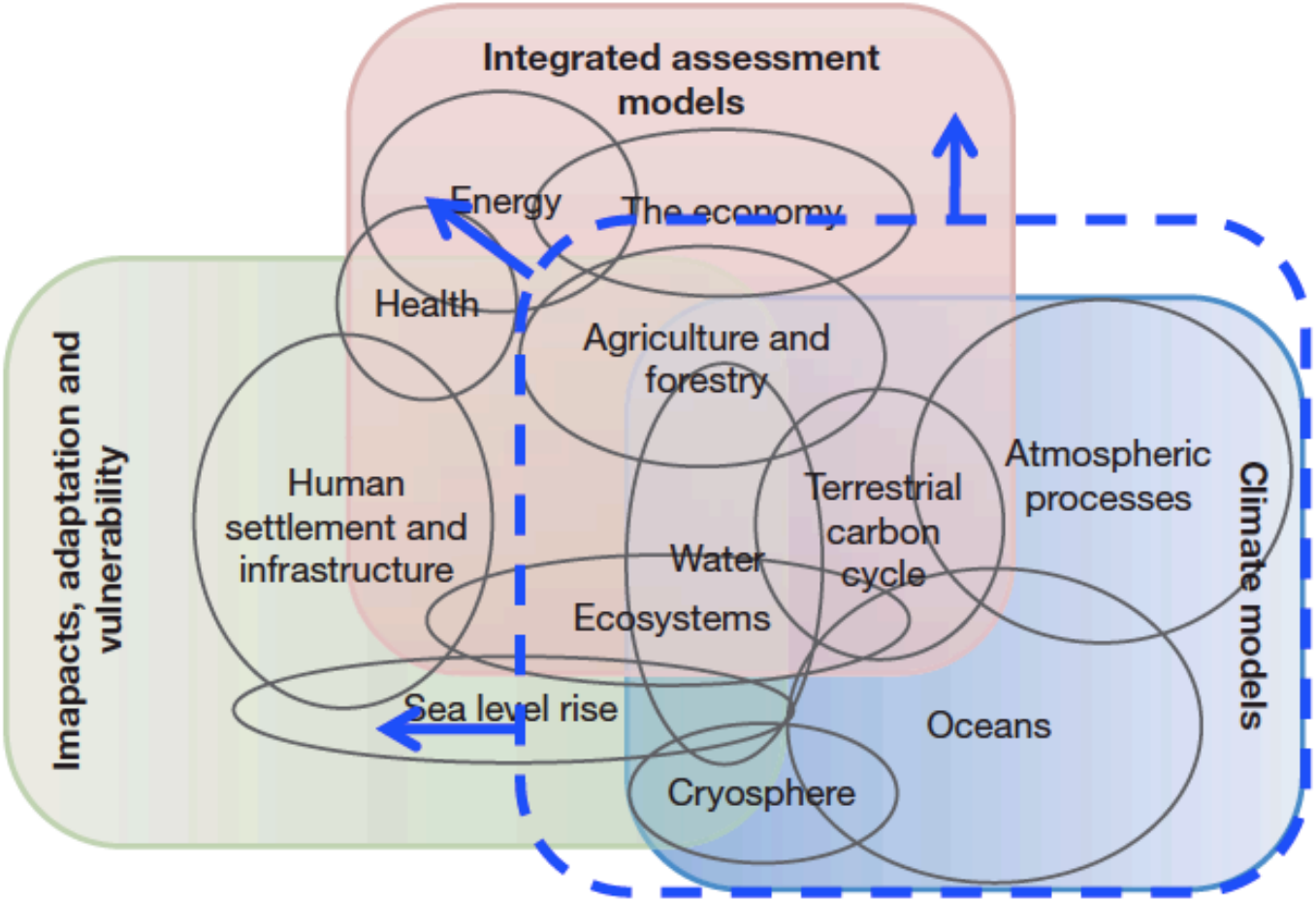
Continuous Development and Improvements:

- Porting
- Performance (performance scaling, memory scaling, throughput, efficiency)
- CESM setup, configure, build, execution support
- Testing

CESM Conceptual Model

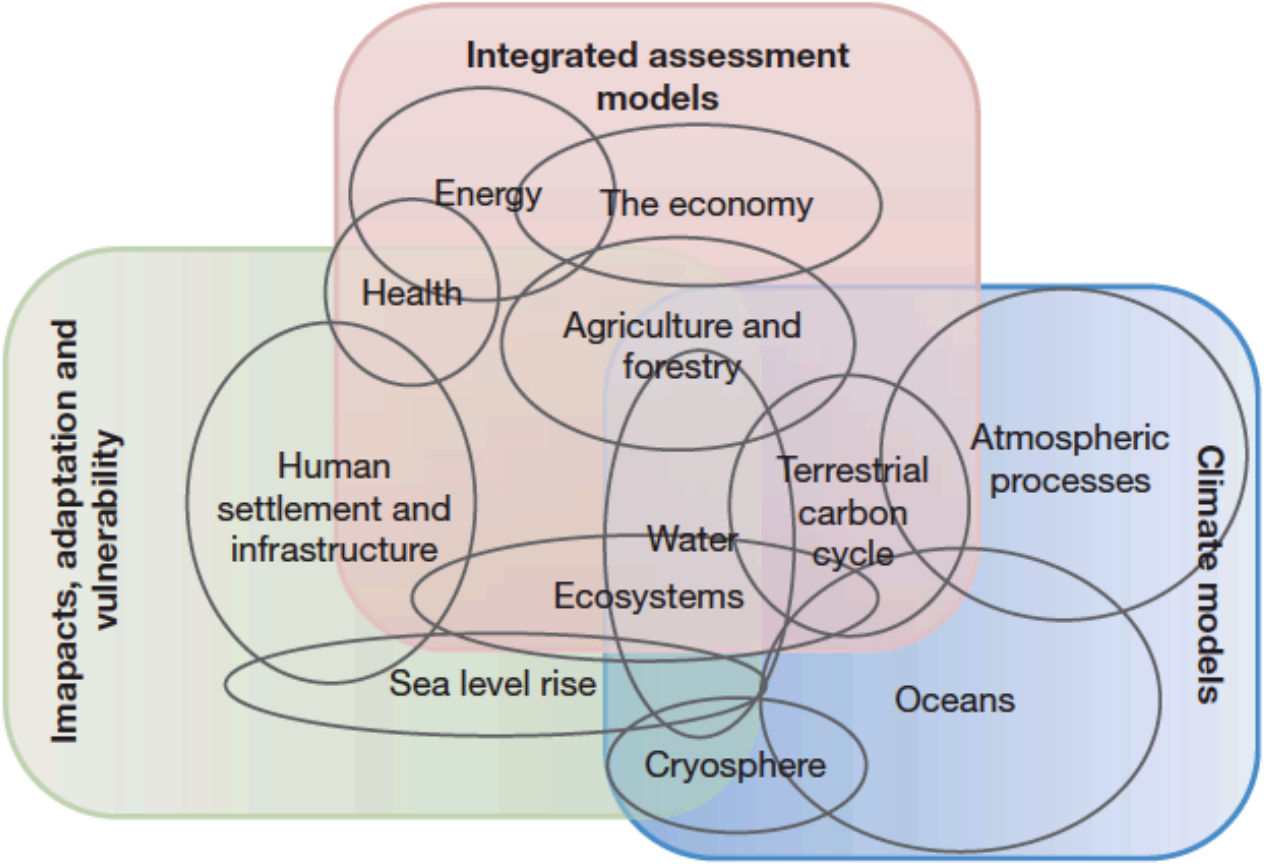


What's an Earth System Model?



http://www.nature.com/nature/journal/v463/n7282/pdf/nature08823.pdf

What's an IAM?



http://www.nature.com/nature/journal/v463/n7282/pdf/nature08823.pdf

Complexity in CESM

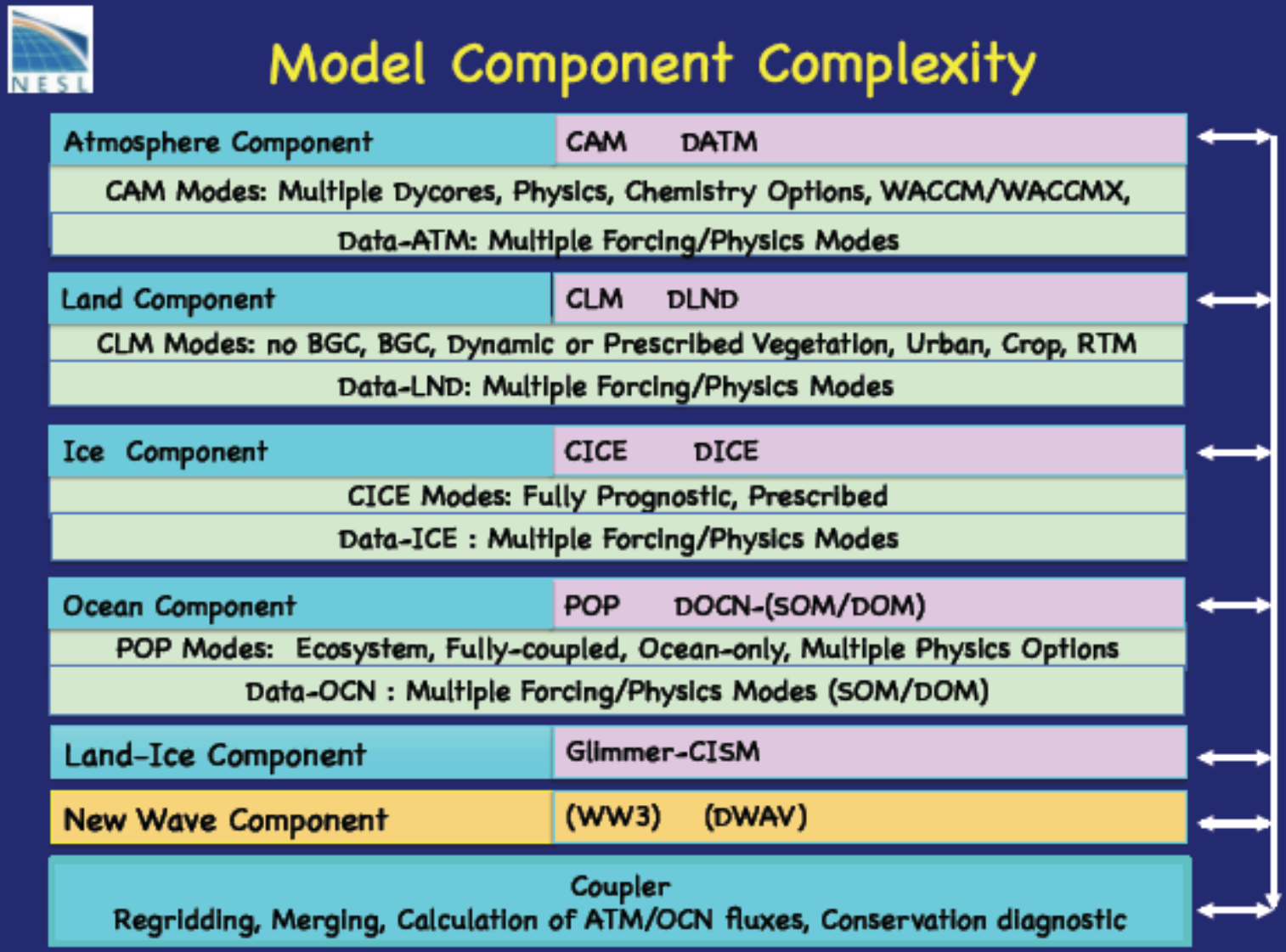
- Simple conceptual models mask enormous complexity
- CESM feature complexity increasing rapidly:
  - Additional component models
  - New physical parameterization options
  - Finer spatial resolution
  - New grids (tripole, cubed sphere, regional grids/mesh refinement, unstructured grids)
  - Support for data assimilation (multiple instances of a component in one run)
- Complexity also arises from:
  - Extensive, diverse developer community (100's of developers)
  - Large number of supported platforms (HPC leading-edge machines)
  - Large code base (>1.3M lines)
  - Large number of input datasets and startup options
  - Large number of supported combinations of options (100 component sets)
  - Need for performance and memory scaling on 1000's of processors

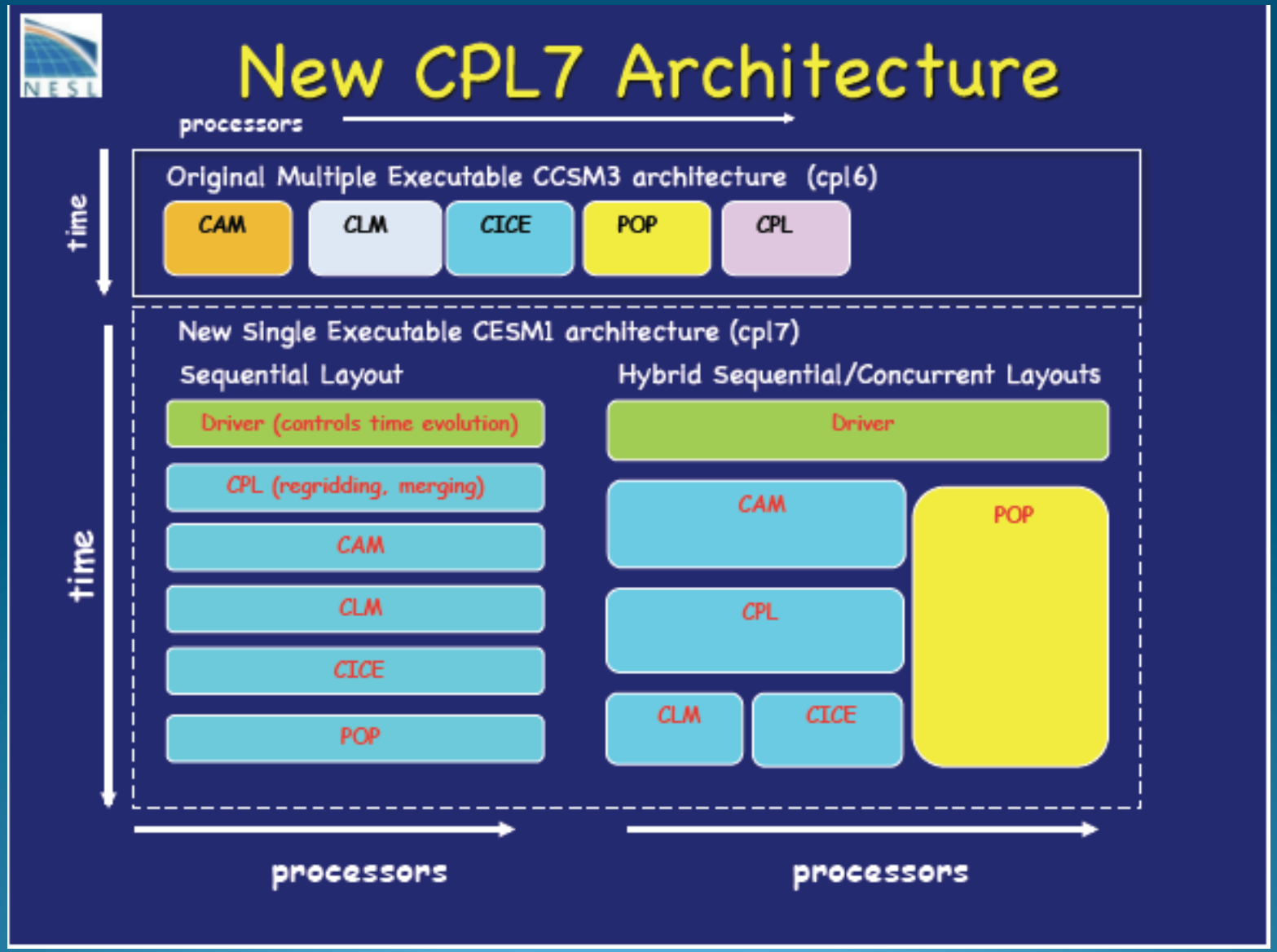
Complexity in CESM

CESM development community is enormous; ***about 450 people*** have developer-level access to the CESM code repository:

- CESM Software Engineering Group (CSEG)
- CESM Working Groups (atmosphere, ice, land, ocean, polar climate, biogeochemistry, chemistry-climate, climate variability and change, paleoclimate, whole-earth, societal dimensions, software engineering)
- Computational and Information Systems Laboratory (CISL)
- Earth System Modeling Framework (ESMF)
- U.S Department of Energy Scientific Discovery through Advanced Computing (SciDAC)

Model Component Complexity




New CPL7 Architecture
processors
time
Original Multiple Executable CCSM3 architecture (cpl6)
CAM
CLM
CICE
POP
CPL
time
New Single Executable CESM1 architecture (cpl7)
Sequential Layout
Driver (controls time evolution)
CPL (regridding, merging)
CAM
CLM
CICE
POP
Hybrid Sequential/Concurrent Layouts
Driver
CAM
CPL
CLM
CICE
POP
processors
processors

Complexity in CESM

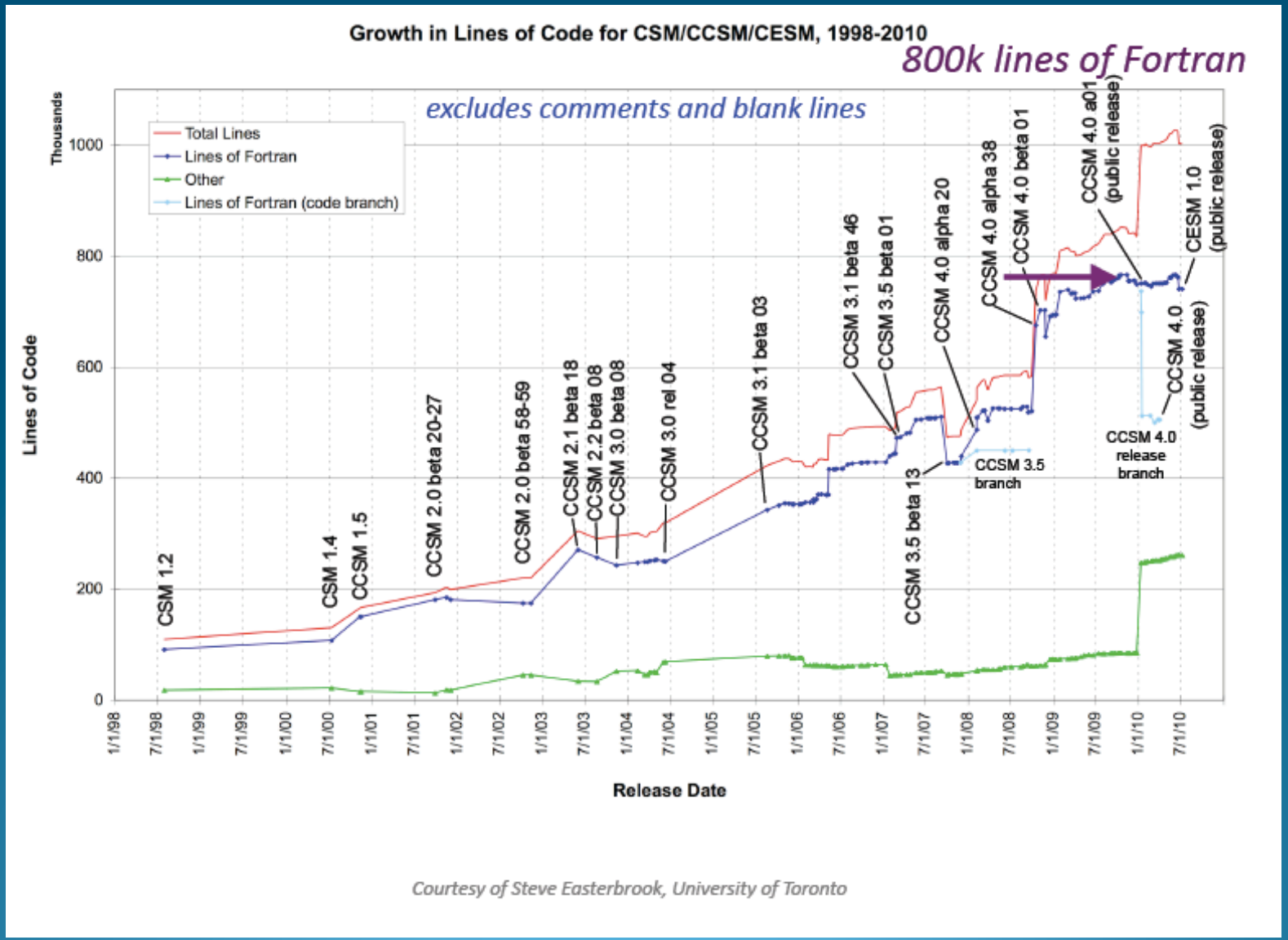
- **229 Configuration Variables**
  - **42 variables for the overall experiment**
  - **58 variables for the "configure" step**
  - **34 variables for the parallel layout**
  - **27 variables for the "build" step**
  - **68 variables for the "run" step**

Information courtesy of James B. White III

Complexity in CESM

- **Initialization Data**
  - **Initial states**
  - **Prescribed quantities (emissions, concentrations, land use)**
  - **2 TB, 9400 files and growing, due to finer resolution and increasing model complexity**
  - **Copies of initialization data available on disk at each supercomputer site**

Information courtesy of James B. White III



Courtesy of Steve Easterbrook, University of Toronto

Complexity in CESM

- **CESM Engineered for Portable Performance**
  - **Block-oriented computation**
  - **Hybrid parallelism**
  - **Modular parallel communication**
  - **Flexible task parallelism**
  - **Modular built-in timers**
    - **portable performance tuning (not dependent on vendor tools)**
    - **configurable level of detail; can be used to load balance components, select tuning parameters for a component model, discover tuning "opportunities" and performance bugs**

Information courtesy of James B. White III

Complexity in CESM

- **Modular Parallel I/O**
  - **Tunable number and location of I/O tasks**
  - **Choices for underlying implementation: netCDF3, netCDF4, pnetCDF, binary**
  - **Potential for asynchronous I/O**
  - **Potential for in-memory checkpoint/restart**

Information courtesy of James B. White III

Complexity in CESM

- **Not all CESM component models originated at NCAR**
  - **Ocean model based on LANL Parallel Ocean Processor (POP)**
  - **Sea-ice model based on LANL CICE**
  - **Land-ice model based on LANL Community Ice-Sheet Model (CISM), which is based upon GLIMMER**
- **Co-development requires additional levels of coordination, another source of complexity in CESM**

What is CSEG?

- CSEG is the CESM Software Engineering Group
  - Mariana Vertenstein, Head
  - Alice Bertini, David Bailey, Tony Craig, Brian Eaton, Jim Edwards, Diane Feddema, Chris Fischer, Brian Kauffman, Erik Kluzek, Andrew Mai, Nancy Norton, Bill Sacks, Sean Santos
  - Size and composition of the group fluctuates depending on funding
  - Not all CSEG members have full-time appointments
- CCR staff working closely with CSEG: Trey White, Gary Strand

CSEG's Role in CESM

- Expedite CESM scientific development and experimentation by making the model more flexible and extensible
- Manage various model releases and associated control runs
- Interact with the entire CESM community to continuously improve model performance and functionality
- Provide support to CESM working groups

CSEG Tasks

- Integrate developments
- Provide easy-to-use, “out-of-the-box” CESM setups
- Manage code repository
- Test
- Manage internal development (provide basis for scientific developments)
- Manage input data repository -- and protect input data integrity
- Run production simulations
- Process output from production simulations
- Manage public releases
- Provide public support
- Port CESM to multiple platforms
- Maintain or improve model efficiency (memory and performance scaling, throughput/load-balancing)

CSEG Software Engineering Support

- **Model Development**
  - **code development: science, performance, grids/resolutions, integration of contributed code, I/O (e.g., parallel I/O)**
- **Code Management**
  - **code repository**
    - **management strategies for code repo usage**
    - **tagging strategies and naming conventions**
    - **component models plus entire CESM system**
- **Scripts Development**
  - **provide mechanisms to support:**
    - **easy-to-use, out-of-the-box setup**
    - **user-friendly selection of model options**
    - **case setup, build, and execution**
    - **archive model output**
- **Porting**
  - support CESM on multiple platforms, environments, operating systems, compilers, etc

CSEG Software Engineering Support

- **Performance**
  - **load balancing, code modifications**
- **Testing**
  - **create scripts, test suite**
  - **run tests**
  - **create snapshot CESM tags**
  - **report testing results**
- **Production Runs**
  - **case specification**
  - **case management**
  - **data management**
  - **case reporting**
- **Public Releases**
  - development-to-public preparation
  - testing
  - documentation

SEA Best Practices

1. Version control
2. Requirements
3. Code reviews
4. Project management
5. Software design and modeling
6. Design reviews
7. Coding standards
8. Code reuse
9. Code refactoring
10. Testing

CESM Software Engineering Best Practices

1. **Version Control/Code Management:**
    - Three Subversion Repositories
        - development repo
            - component-model trunks and branches
            - trunk_tags, branch_tags
            - naming conventions for branches, tags
        - public-release repo
        - input-dataset repo
    - CESM Tagging Strategy
        - CESM tag is a “tag of tags”
        - low-tech, workable “planned tags” page updated by developers
        - tag frozen, new tag “sandbox” assembled, testing begins, tag created
        - sequence of alpha tags, followed by a beta tag
        - tags webpage with individual tag info, testing record
        - beta tags are checked out onto disk to support scientific community use

CESM Software Engineering "Best Practices"

2. **Requirements**
   - output filename conventions
   - experiment naming conventions
   - input dataset names
   - tried more formal requirements in mid 2000's
   - less formal requirements created intermittently (wiki)
3. **Code Reviews**
   - component-model code reviews handled at the component-model level; common in the OMWG
   - CESM code reviews are rare
4. **Project Management**
   - multiple, simultaneous projects
   - not all projects under CSEG control
   - few formally identified CSEG projects (exceptions: data-model refactoring, CCSM scripts refactoring, others?)

CESM Software Engineering "Best Practices"

5. **Software design and modeling**
   - component models manage their own software design
   - two component models have developer's guides covering design, design reviews, coding standards, etc

6. **Design reviews**
   - component models manage their own design reviews
   - design reviews in CESM rarely happen on a group-wide basis

7. **Coding standards**
   - component models have their own coding standards
   - some coding standards for CESM as a whole

CESM Software Engineering "Best Practices"

8. **Code Reuse**
   - shared code used in data models and active models, for coupler interfaces, file-reading, common physical constants, and orbital information
   - common libraries for I/O, communications
   - shared code provides efficiency and consistency

9. **Code Refactoring**
   - data models (data7 project)
   - CCSM3 scripts (build, configuration, templates, archiving)
   - CESM scripts continuous improvement plan

CESM Software Engineering "Best Practices"

10. Development Code Testing:

- extensive, semi-automated test suite
  - 100's of tests on 5-10 machines
  - good test coverage of supported option combinations
  - efficient reuse of runs for multiple tests (tradeoff is complexity)
  - comprised of "home-grown" scripts
  - exact restarts required (bit-for-bit)
  - test results reported on tags webpage
- alpha test sequence, followed by beta-tag creation
- typically several beta tags per month
- test strategy recently revised; CESM Alpha and Beta Tags and Test Strategy document
- individual component model have their own test suites; some more sophisticated than others
- some unit tests

CESM Software Engineering "Best Practices"

10. Release Testing

- Separate testing of Releases
- Release checklist
- About a one-month testing period between last modification to public release
- "vendor release" of CCSM4

More CESM Software Engineering "Best Practices"

- **Public Support**
  - extensive documentation for CESM and for all component models
  - community tutorials
    - presentations at CESM annual workshops
    - 1-week new-user tutorial at NCAR
  - bulletin board
  - cesm-help@cgd.ucar.edu
  - performance table (load balancing, throughput, charging info)
- **Defect Tracking**
  - Bugzilla
  - inconsistent usage; rarely reviewed by CSEG
- **Climate Simulation Management**
  - simple, "ticket" system for fully specifying each production run (home-grown, text-based)
  - *run database will soon replace this simple system* (developed by Alice Bertini; future SEA talk?)

Barriers to CESM Best Practices

- **Factors not under direct CSEG control**
  - Time pressures
  - Funding pressures
    - positions eliminated or part-time assignments
    - conflicting priorities
    - heavy workloads
  - CSEG is not co-located
- **Factors under our control**
  - Expediency
  - Communication breakdowns
  - Training

Better CESM Best Practices?

- **CSEG-identified Issues:**
  - Prioritizing development vs support for the released product
  - Better user interface?
  - More/better documentation on procedures for community and developers
  - Better view of the "bigger picture" and future developments
  - Better follow-through on policies/plans/conventions
  - Testing coverage is at the macro level; could benefit from testing at the micro-level. Unit testing is rare, and difficult to retrofit into existing designs.
  - More frequent design reviews

Saving the Best for Last:

Better Best Practices and Means of Achieving Them

- Advice?
- Discussion on Improvements
  - Version control
  - Requirements
  - Code reviews
  - Project management
  - Software design and modeling
  - Design reviews
  - Coding standards
  - Code reuse
  - Code refactoring
  - Testing