NEMS FV3GFS Community Modeling System Training and Tutorial: Planning and Preparation Meeting
19-20 July @GFDL Rm 217

Support slides for 1030 - NEMS ESMF & NUOPC Mediator (Mark Iredell)

(Most of these slides were developed by Rocky Dunlap and Fei Liu for NEMS training given at EMC in November 2016. Formatting mishaps are mine when copying to this google slides file.)

What is NEMS?

The **NOAA Environmental Modeling System (NEMS)** is a unified modeling system that supports multiple coupled modeling applications.

Each **coupled modeling application** is associated with a purpose, a set of model components, and a range of supported options including grids and resolutions.

A **unified infrastructure** means that many parts of NEMS are shared across modeling applications, including the build system, the coupling infrastructure, mediators, testing infrastructure, a common way of representing run configurations, and a common way to execute configurations.

Some Forces Affecting NEMS Design

- Increase in number of modeling component types
  - e.g., atmosphere, ocean, ice, land, hydrology, wave, ionosphere
- Multiple options for each modeling component type
  - e.g., MOM5/6, HYCOM, “data” component
- Community components developed external to EMC
- Multiple configurations needed for controlled experimentation
  - standalone model
  - coupled with “data” component
  - coupled with limited feedbacks
  - fully coupled
  - ensembles
  - alternative parameterizations
  - different resolutions

NEMS Modeling Applications

Modeling Application	ATM	OCN	ICE	WAV	LND	AER	HYD	ION	CST
UGCS-Weather	o	o	o	o	o	o		o	o
UGCS-SubSeasonal	o	o	o	o	o	o		o	o
UGCS-Seasonal	o	o	o	o	o	o		o	o
WAM-IPE	o							o	
HYCOM-Ice (RTOFS)		o	o						
Wave Prediction	o			o					
Regional	o	o	o		o		o		
Regional Nest	o	o							
Regional Arctic	o	o	o						
CMAQ Air Quality	o					o			

This is a simplified version of the NEMS application spreadsheet. The spreadsheet includes interim milestones for each application.

Earth System Modeling Framework

The Earth System Modeling Framework (ESMF) was initiated in 2002 as a multi-agency response to calls for common modeling infrastructure.

ESMF provides:

- high performance utilities, including grid remapping, data communications, and model time management
- an component-based architecture for model construction

ESMF Metrics:
~7000 downloads

~100 components in use

~3000 individuals on info mailing list

~40 platform/compilers regression tested nightly

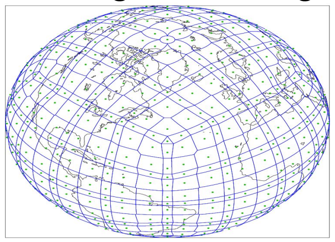
~6500 regression tests

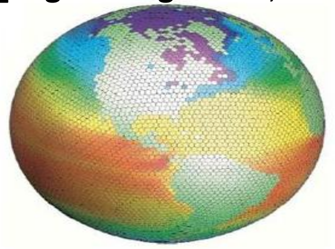
~1M SLOC

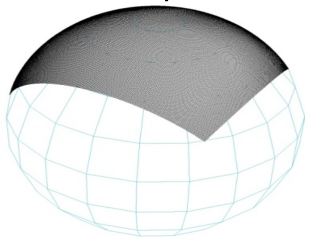
NUOPC is a software layer on top of ESMF that ensures interoperability of components. Most major U.S. modeling centers have adopted NUOPC conventions.

ESMF Grid Remapping

- Uniquely fast, reliable, and general – interpolation weights computed in parallel in 3D space
- Supported grids:
  - Logically rectangular and unstructured grids in 2D and 3D, point clouds/observations
  - Global and regional grids
- Supported interpolation methods:
  - Nearest neighbor, bilinear, higher order patch recovery, and 1st order conservative methods
  - Options for straight or great circle lines, masking, and a variety of pole treatments
- Multiple ways to call ESMF grid remapping:
  - Generate and apply weights using the **ESMF API**, within a model
  - Generate and apply weights using **ESMPy**, through a Python interface
  - Generate weights from grid files using **ESMF_RegridWeightGen** , a command-line utility


HOMME Cubed Sphere Grid with Pentagons
Courtesy Mark Taylor of Sandia


FIM Unstructured Grid


Regional Grid

Standard Component Interfaces

ESMF/NUOPC components have three kinds of methods with standard interfaces:

- Initialize, Run, and Finalize

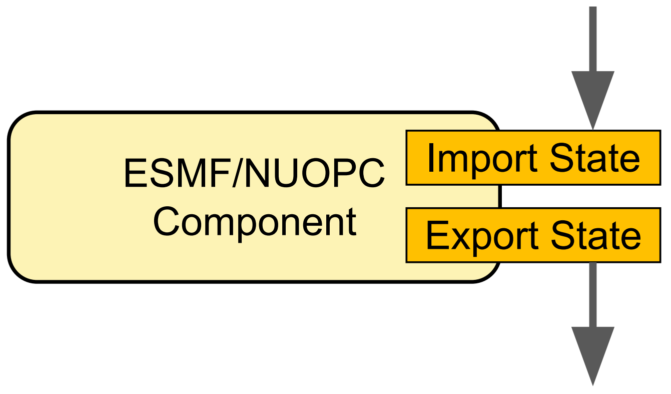
```
subroutine InitP1(comp, importState, exportState, clock, rc)
  type(ESMF_GridComp)    :: comp
  type(ESMF_State)       :: importState
  type(ESMF_State)       :: exportState
  type(ESMF_Clock)       :: clock
  integer, intent(out)   :: rc

  ! This is where the model specific setup code goes.
  rc = ESMF_SUCCESS

end subroutine InitP1
```

Interfaces are wrappers and can often be introduced in a non-intrusive and high-performance way. ESMF is designed to coexist with native model infrastructure.

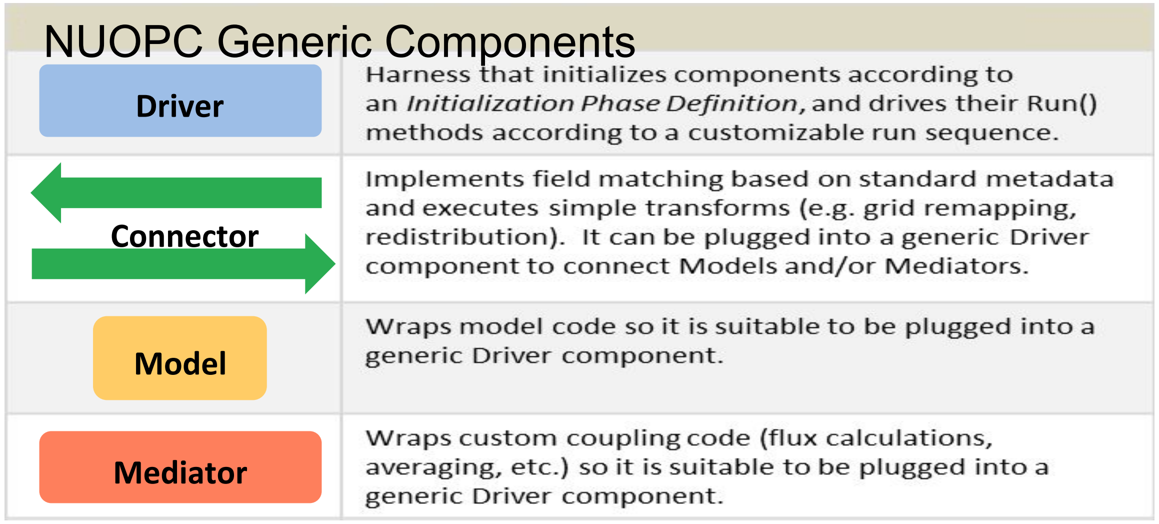
Components Share Data via Import and Export States



- **Components do not directly access each other's data**.
- The only way data moves in or out of a Component is via instances of the ESMF **State** class (`ESMF_State`).
- A State is a **container** for ESMF data types that wrap native model data.
- Model data can be **referenced**, avoiding duplicates and copies.
- **Metadata** (e.g., name, coordinates, decomposition) travels with data objects.

NUOPC Layer Components

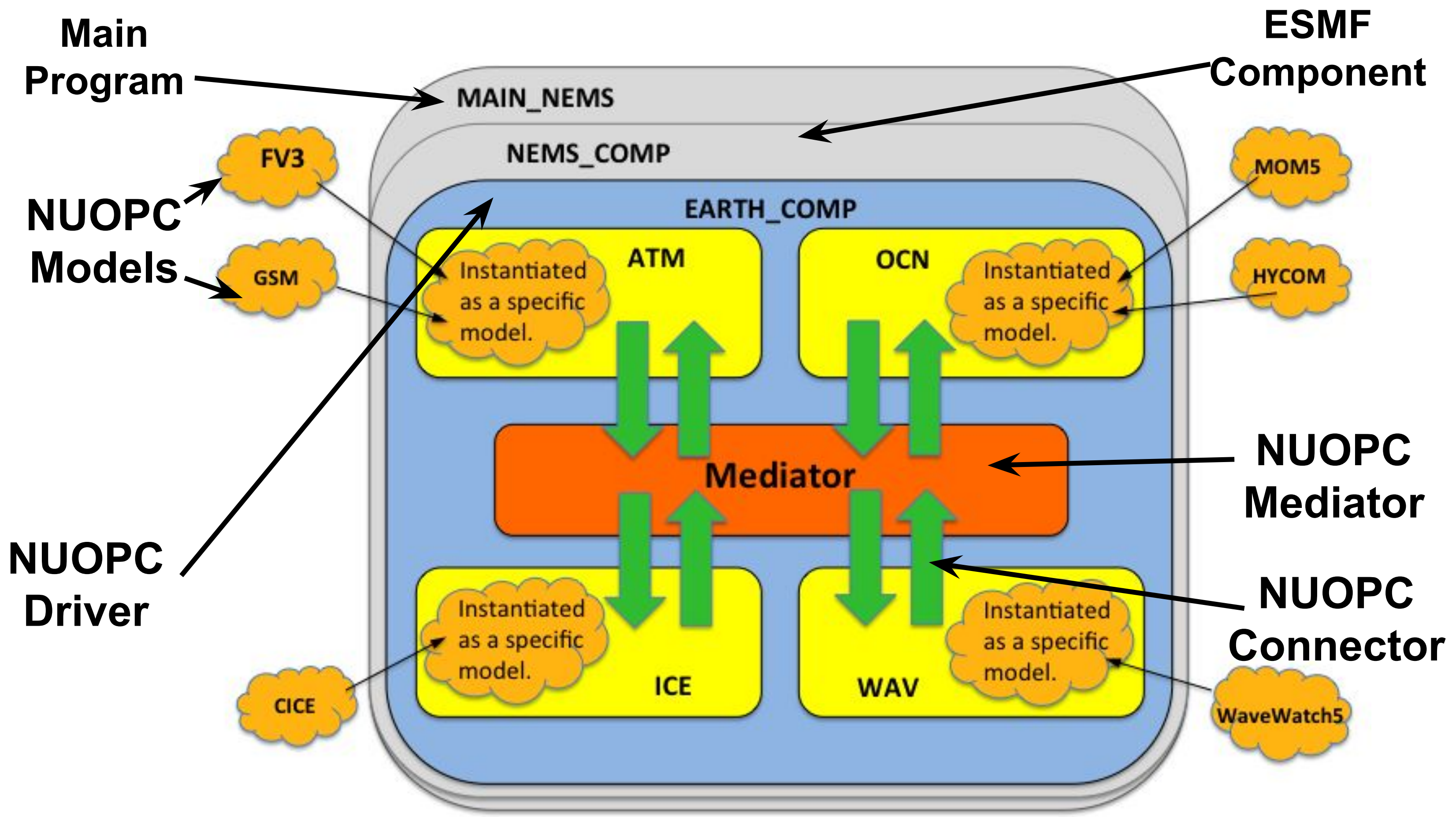
The NUOPC Layer’s four **generic components** represent the major ştructural pieces needed to build coupled models.



NUOPC Generic Components

Driver	Harness that initializes components according to an *Initialization Phase Definition*, and drives their Run() methods according to a customizable run sequence.
Connector	Implements field matching based on standard metadata and executes simple transforms (e.g. grid remapping, redistribution). It can be plugged into a generic Driver component to connect Models and/or Mediators.
Model	Wraps model code so it is suitable to be plugged into a generic Driver component.
Mediator	Wraps custom coupling code (flux calculations, averaging, etc.) so it is suitable to be plugged into a generic Driver component.

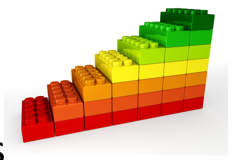
From Theurich et al. 2016


Main Program
ESMF Component
MAIN_NEMS
NEMS_COMP
EARTH_COMP
NUOPC Models
FV3
GSM
MOM5
HYCOM
ATM
OCN
Instantiated as a specific model.
Mediator
NUOPC Mediator
NUOPC Driver
NUOPC Connector
ICE
WAV
CICE
WaveWatch5

Component Interoperability

How does NUOPC provide interoperability?

- Each component has **a single, public entry point** called `SetServices`
- A **standard initialization sequence** to:
  - – link coupling fields across models
  - – synchronize clocks
  - – set initial conditions



- A set of **standard data types** for representing model grids and fields
- **Standard names** for coupling fields
- A **standard run phase** that ensures that incoming and outgoing fields are consistent with the component's clock.
- A **makefile fragment** with a small number of variables used for compiling and linking against the component.

NUOPC Initialization Sequence

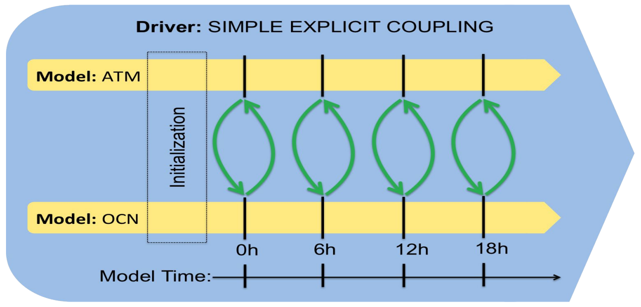
The Initialize Phase Definition (IPD) defines an initialization **handshake** among all components in an application.

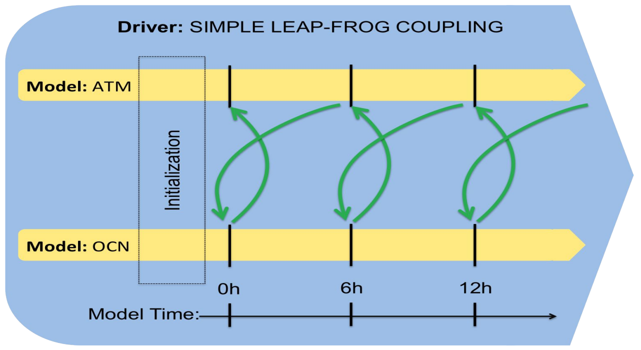
The initialization sequence ensures that:

- Each Model advertises a set of required import and available export Fields and each has a standard name.
- Each Connector matches Fields between Models and establishes a communication RouteHandle.
- All Models have their import Field dependencies satisfied.
- All Model Clock’s agree on the start time.
- Two Models with connected Fields agree on who will provide the grid structure (can be both).
- Model Fields are initialized at the correct time and a timestamped according to the Model clock.
- The Driver has established a run sequence.

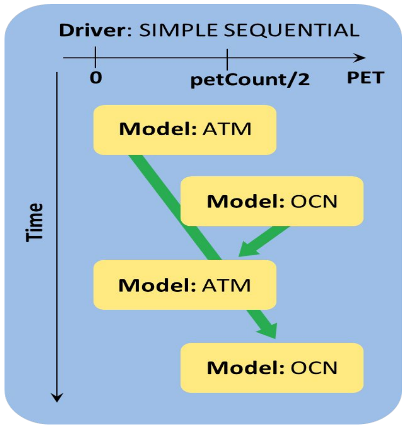
Run Sequence

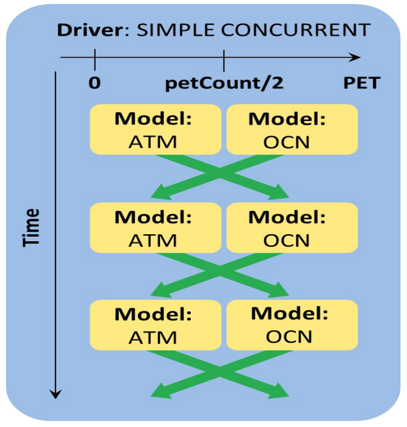
- Each Driver maintains a Run Sequence over its child Components.
- The Run Sequence is **set dynamically** (during the initialize sequence) and controls the master time loop.
- Run Sequences supported:
  - **Simple loops** in which all Components exchange data at the same coupling interval
  - **Multiple timescales** in which some Components communicate less frequently than others
  - **Explicit, semi-implicit, and implicit modes** in which some Components may run ahead or lag behind to satisfy numerical constraints





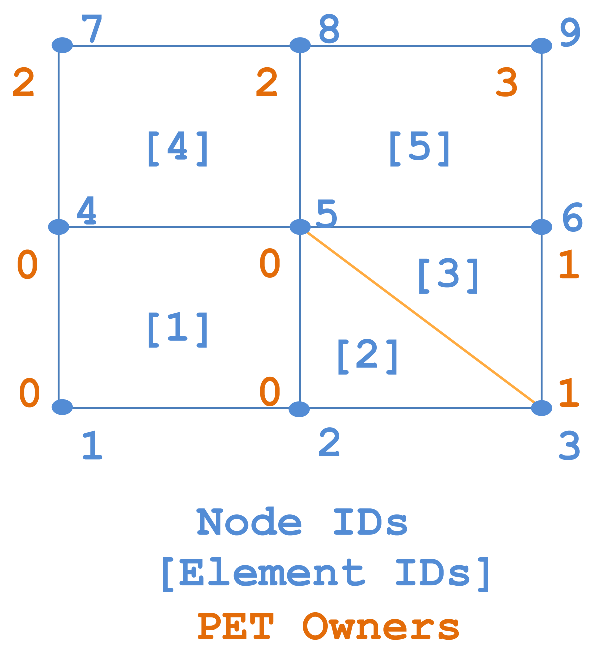
Sequential and Concurrent Components





- Drivers support **both sequential and concurrent execution** of child Components.
- The amount of concurrency is dependent on:
  - whether Components are assigned disjoint or overlapping PETs
  - data dependencies introduced by scientific constraints, e.g., the frequency of coupling required
- Drivers have the option of assigning a PET list when a child component is added (Model or Mediator)

ESMF Meshes



- A Mesh is constructed of **nodes** and **elements**.
- The **parametric dimension** is the dimension of the elements.
- The **spatial dimension** is the dimension of the space the Mesh is embedded in, i.e., the number of coordinate dimensions.
- ESMF support 2D element in 2D space, 3D elements in 3D space, and 2D element in 3D space.
  - In 2D, there is no limit to the number of polygon sides.
  - In 3D, elements may be tetrahedra or hexahedra.
- Meshes are distributed by element. Nodes may be duplicated on multiple PETs but are owned by one PET.

ESMF Regridding

Regridding (or remapping or interpolation) is the process of moving data from one grid to another while preserving qualities of the original data.

- **Flexible**:
  - Computes weights between a wide range of grids: structured and unstructured, global and regional, 2D and 3D, spherical and Cartesian
  - Options for interpolation method, pole treatment, masked points, …
- **Well Tested and Portable** :
  - >200 regridding cases tested every night
  - >40 different OS/Compiler/MPI combinations tested every night
- **Parallel and Fast** :
  - Able to compute weights in minutes which before took hours
  - Able to compute weights between very large grids
- **Community developed** :
  - Supported by NASA, NOAA, DOD and NSF funding
  - Well established (since 2005) community processes for prioritization, support and review, priorities set through quarterly Change Review Board (CRB) meetings

NGGPS Schematic

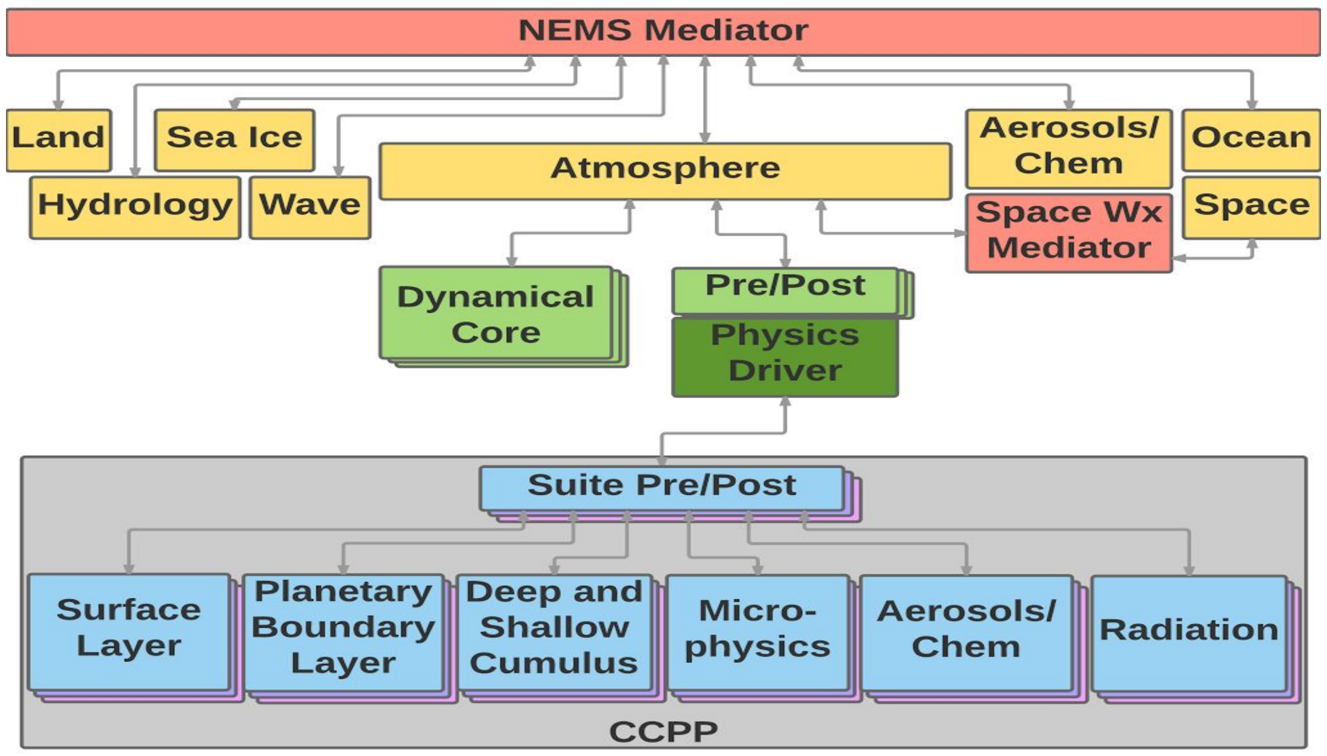
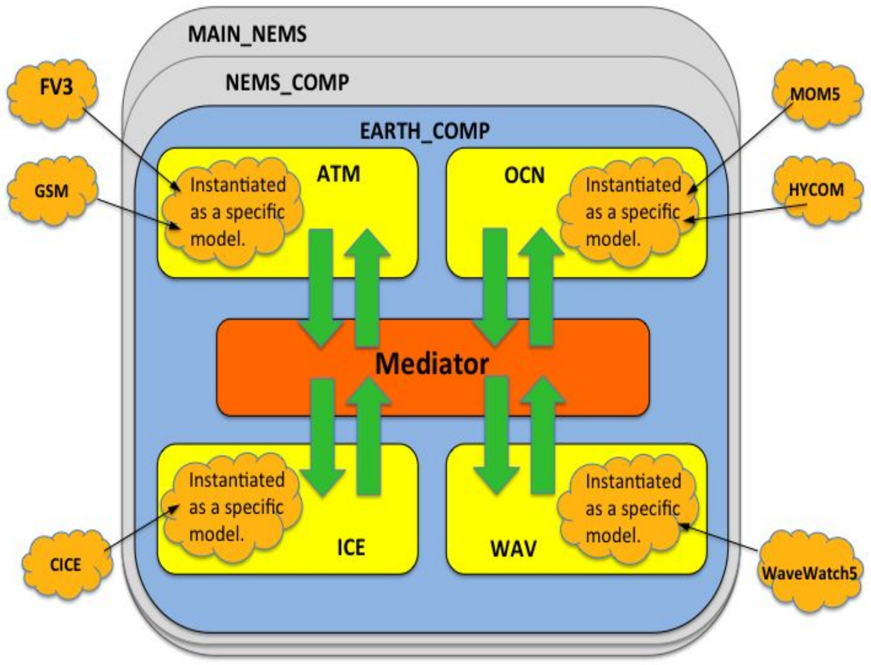


Image courtesy of the Developmental Testbed Center

Overview of Four NUOPC Component Types in NEMS

- All model components have a NUOPC Model “cap.”
- There is a NUOPC Driver.
- There is a main NUOPC Mediator.
- NUOPC Connectors perform parallel data transfers and regridding operations.

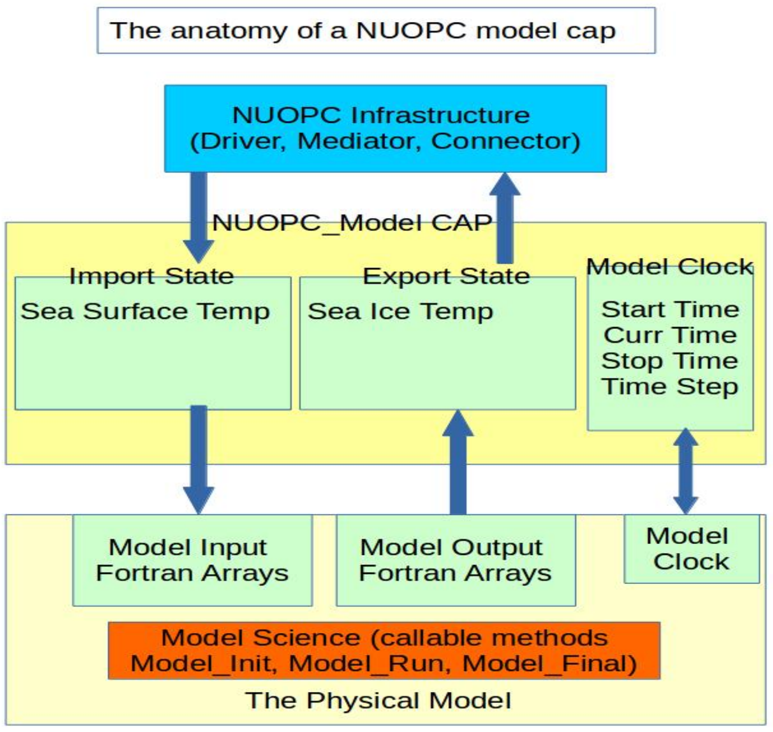

MAIN_NEMS
NEMS_COMP
EARTH_COMP
FV3
GSM
MOM5
HYCOM
ATM
OCN
Instantiated as a specific model.
Instantiated as a specific model.
Mediator
Instantiated as a specific model.
Instantiated as a specific model.
ICE
WAV
CICE
WaveWatch5

NUOPC “Caps”

In a NUOPC application, all components **interact with other components through NUOPC “caps.”**

These are wrappers that translate native model time, grids, and memory layouts into standard forms that the framework can understand.

The **same** “cap” source code is used in multiple applications.



The anatomy of a NUOPC model cap
NUOPC Infrastructure (Driver, Mediator, Connector)
NUOPC_Model CAP
Import State
Sea Surface Temp
Export State
Sea Ice Temp
Model Clock
Start Time
Curr Time
Stop Time
Time Step
Model Input Fortran Arrays
Model Output Fortran Arrays
Model Clock
Model Science (callable methods Model_Init, Model_Run, Model_Final)
The Physical Model

Role of the NUOPC Driver in NEMS

The NEMS **NUOPC Driver controls all Models, Mediators, and Connectors** in a NEMS application.

The Driver's responsibilities are:

- to register the Mediator, all selected Models, and Connectors as child components
- assign processor resources to each child component
- drive each component through the initialization sequence
- set up a run sequence, i.e., the order of execution of all child components

- execute the run sequence

$\Big\}$ initialization sequence

Sample NEMS Configure File (nems.configure)

```
##################################
# NEMS Run Time Configuration File  #
##################################
# EARTH #
EARTH_component_list: MED ATM OCN ICE

# MED #
med_model:              nems
med_petlist_bounds:     60 65

#ATM#
atm_model:             gsm
atm_petlist_bounds:     0 31

# OCN #
ocn_model:
ocn_petlist_bounds:

# ICE #
ice_model:                        cice
ice_petlist_bounds:     56 59
```

Run sequence ingested into Driver

```
# Run Sequence #
runSeq::
 @1800.0
  MED MedPhase_prep_ocn
  MED -> OCN :remapMethod=redist
  OCN
  @600.0
   MED MedPhase_prep_ice
   MED MedPhase_prep_atm
   MED -> ATM :remapMethod=redist
   MED -> ICE :remapMethod=redist
   ATM
   ICE
   ATM -> MED :remapMethod=redist
   ICE -> MED :remapMethod=redist
   MED MedPhase_atm_ocn_flux
   MED MedPhase_accum_fast
  @
  OCN -> MED :remapMethod=redist
 @
::
```

Colors show actions performed by:

- Connectors (->)
- Mediator (MED)
- Models

(@) indicates coupling interval

NEMS Driver Code (module_EARTH_GRID_COMP.F90)

- Conditionally imports (`#ifdef` around `USE` statement) NUOPC Model caps for ATM, OCN, ICE, WAV, LND, HYD, etc.

- `EarthRegister()`
  - Loads *nems.configure* file
  - Registers methods below

- `SetModelServices()`
  - Dynamically adds child models, mediator, and connectors to the driver (via `NUOPC_DriverAddComp()`)
  - Sets PET (processor) list for each child
  - Configured by *nems.configure*

- `SetRunSequence()`
  - Dynamically sets order of execution of models, mediator, and connectors
  - Configured by *nems.configure*

Role of the NEMS Mediator

- The Mediator is set up with ATM, OCN, ICE, LND, and HYD components.
- Slow (ocean) and fast (atmosphere and ice) coupling phases
- The mediator includes the following functions:
    - Connects fields whose standard names match
    - Accumulates and averages atmosphere and ice fields between calls to the ocean model
    - Merges fields with a generic merge method that allows for weighting
    - Performs custom coupling operations, along with unit transformations
    - Performs interpolation (fluxes are mapped conservatively, states bilinearly, higher order also available)

More information about the mediator:
http://cog-esgf.esrl.noaa.gov/projects/couplednems/mediator_design

Worksheet of planned coupling fields across all modeling applications:
https://docs.google.com/spreadsheets/d/11t0TqbYfEqH7ImTZ7dYe1DSCh6vOUFgX-3qvXgce-q0/edit#gid=0

Thanks