Code Review:

NUOPC in NEMS for OCEAN coupling

Gerhard Theurich, NESII
Remarks

- Introduce NUOPC Layer concepts by looking at the current NEMS code.
- Code is as of revision 33333 on https://svnemc.ncep.noaa.gov/projects/nems/trunk/
- Will be switching between slides and code a few times.
- Important: This is work in progress! We are looking at a snapshot!
NUOPC Background

- **National Unified Operational Prediction Capability**
  - Consortium of U.S. operational weather and water prediction centers.

- Participants: NOAA, Navy, Air Force, NASA, and other associated modeling groups.

- Develop a Common Model Architecture (CMA) to:
  - Improve collaboration among agencies.
  - Accelerate the transition of new technology into the operational centers.

- NUOPC websites:
  - [http://www.weather.gov/nuopc/](http://www.weather.gov/nuopc/)
  → NUOPC Layer Reference Manual + NEMS Plan Roadmap
Writing a NUOPC compliant ESMF component the “easy way” by using the **NUOPC Layer**:
- Software layer implemented on top of ESMF.
- Generic components (Model, Mediator, Connector, and Driver) to derive from; Specialize by attachable methods.
- Utility methods for common tasks.
- Field dictionary (standard names and units) for matching.

**Compliance checker option, as a development & debugging tool.**

**Part of ESMF; v6.2.0 is the current NUOPC Layer reference release. Using v6.3.0 snapshots in NEMS.**
Basic Building Blocks

**Model:**
- Typically implements a specific physical domain, e.g. atmosphere, ocean, ice.

**Connector:**
- Connects pairs of components in one direction, e.g. Model to/from Model, or Model to/from Mediator.
- Executes simple transforms (Regrid or Redist).

**Mediator:**
- Used for custom coupling code (flux calculations, averaging, etc.) between Models.

**Driver:**
- Provides a harness for Models, Mediators, and Connectors.
- Coordinates initialize and run sequences.
Architectural Options

Driver: SIMPLE
Model: ATM
Model: OCN

Driver: COUPLED WAVE
Model: ATM
Model: ICE
Model: OCN
Model: WAVE

Driver: INTERACTIVE ENSEMBLE
Model: ATM
Driver: ATM ENSEMBLE
Model: ATM_1
Model: ATM_2
Model: ATM_3
Model: ATM_4

Table:

<table>
<thead>
<tr>
<th></th>
<th>ICE</th>
<th>OCN</th>
<th>LND</th>
<th>MED</th>
<th>ATM1</th>
<th>ATM2</th>
<th>ATM3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATM3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Current NEMS Architecture based on NUOPC

- **ATM**
  - Send instantaneous and averaged Fields
  - Receive Instantaneous Fields
  - Instantiated as a specific model.

- **OCN**
  - Send Instantaneous Fields
  - Receive Instantaneous and averaged Fields

- **Mediator**
  - Continue the averaging of averaged Fields
  - Forward instantaneous Fields

- **EARTH_COMP**
  - Med2atm
  - Ocn2med

- **MAIN_NEMS**
  - **NEMS_COMP**

- **MOM5**
  - HYCOM
  - DummyOCN

Legend:
- ATM
- OCN
- MED

Directional Arrows:
- ATM to OCN
- OCN to ATM
- MED to ATM
- MED to OCN
- OCN to MED
• Look at

    module_EARTH_GRID_COMP.F90

• Set petLists for ATM, OCN, MED according to run-time configuration (nems.configure).

• Select specific ATM and OCN models according to build- (configure.nems) and run-time (nems.configure) configuration.

• Plug a specific Mediator into the EARTH Driver.

• Define the connection matrix.
# NEMS Configuration File#

# ATM #
atm_model: gsm
atm_petlist_bounds: 0 31

# OCN #
ocn_model: dummy
ocn_petlist_bounds: 32 49

# MED #
med_petlist_bounds: 50 63
med_atm_coupling_interval_sec: 7200.0
med_ocn_coupling_interval_sec: 21600.0
Current NEMS RunSequence with NUOPC
• ATM Fields:

IMPORT: sea_surface_temperature (instant)
EXPORT: surface_net_downward_shortwave_flux (average)
          air_temperature_at_lowest_level (instant)

• MED Fields:

IMPORT: sea_surface_temperature (instant)
          surface_net_downward_shortwave_flux (average)
          air_temperature_at_lowest_level (instant)
EXPORT: sea_surface_temperature (instant)
          surface_net_downward_shortwave_flux (average)
          air_temperature_at_lowest_level (instant)

• OCN Fields:

IMPORT: surface_net_downward_shortwave_flux (average)
          air_temperature_at_lowest_level (instant)
EXPORT: sea_surface_temperature (instant)
Specialization of Generic Components

- Going beyond “Initialize/Run/Finalize”
- Balance the need for additional complexity (for negotiation, compatibility checking, self-description) with the resulting burden on the developer.
- Generic components with clearly defined specialization points: a well established OOP pattern: “inheritance and abstract classes”
- Developer is guided to specific aspects that need to be implemented.
- Concept of application code called by the library is the basis of ESMF components; NUOPC Layer goes one step further.
Application

```fortran
... call CompInit()
... end
```

```fortran
subroutine app_special()
... end subroutine
```

ESMF

```fortran
subroutine CompInit()
... call app_special()
... end subroutine
```
subroutine CompInit()
  ... call generic_CompInit()
  ...
end subroutine

subroutine generic_CompInit()
  ...
  ! generic code, e.g. checking
  call app_CompInit_point1()
  ...
  ! more generic code
  call app_CompInit_point2()
  ...
  ! generic code, e.g. time stamp
end subroutine

subroutine app_CompInit_point1()
  ...
  ! special code that goes
  ! beyond generic NUOPC code
end subroutine
NUOPC Layer Features

• Architectural choices
  - Parametrized in a Driver component.

• Field brokering
  - Field dictionary based on CF "standard name" metadata.

• Initialize sequence
  - Different levels are implemented.
  - Field brokering between producer and consumer components.
  - Resolving data dependencies between components during initialize.

• Run sequence
  - Parametrized run sequence.
  - Fully customizable (explicit, semi-implicit, implicit schemes).
  - Component based (run components, not code directly)
  - Support for multiple time scales.