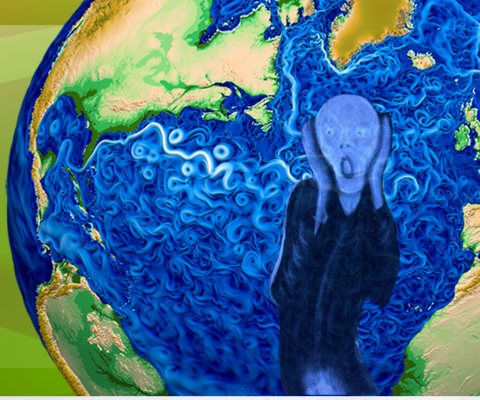
E3SM
Energy Exascale
Earth System Model



Design of a Next-Generation Atmospheric Driver for SCREAM

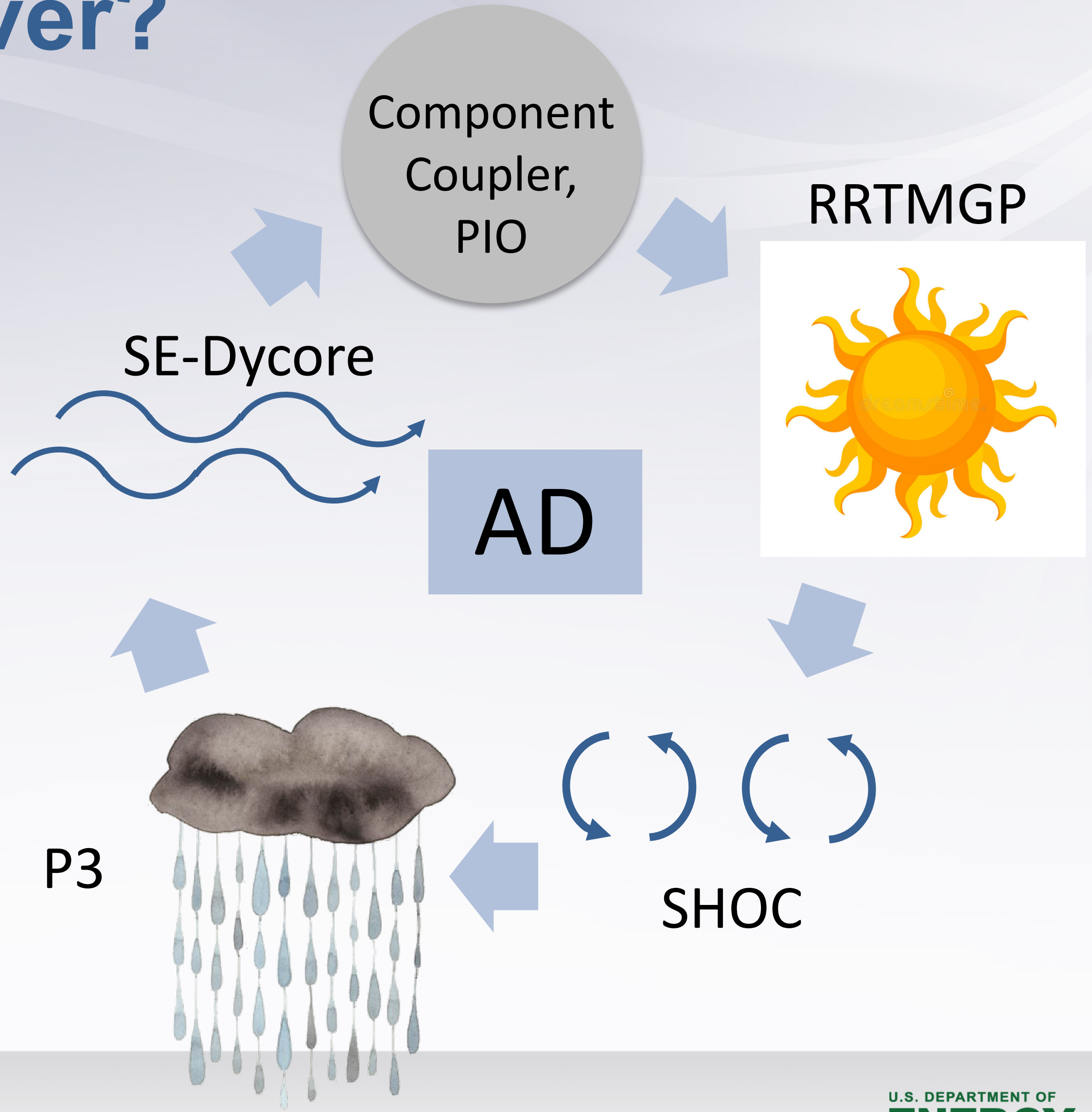
Aaron Donahue and Luca Bertagna

E3SM All-Hands, March 19-21, 2019

Lawrence Livermore National Laboratory
Pacific Northwest National Laboratory
Sandia National Laboratories
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 IM Release Number LLNL-PRES-769384
U.S. DEPARTMENT OF ENERGY

What is the Atmospheric Driver?

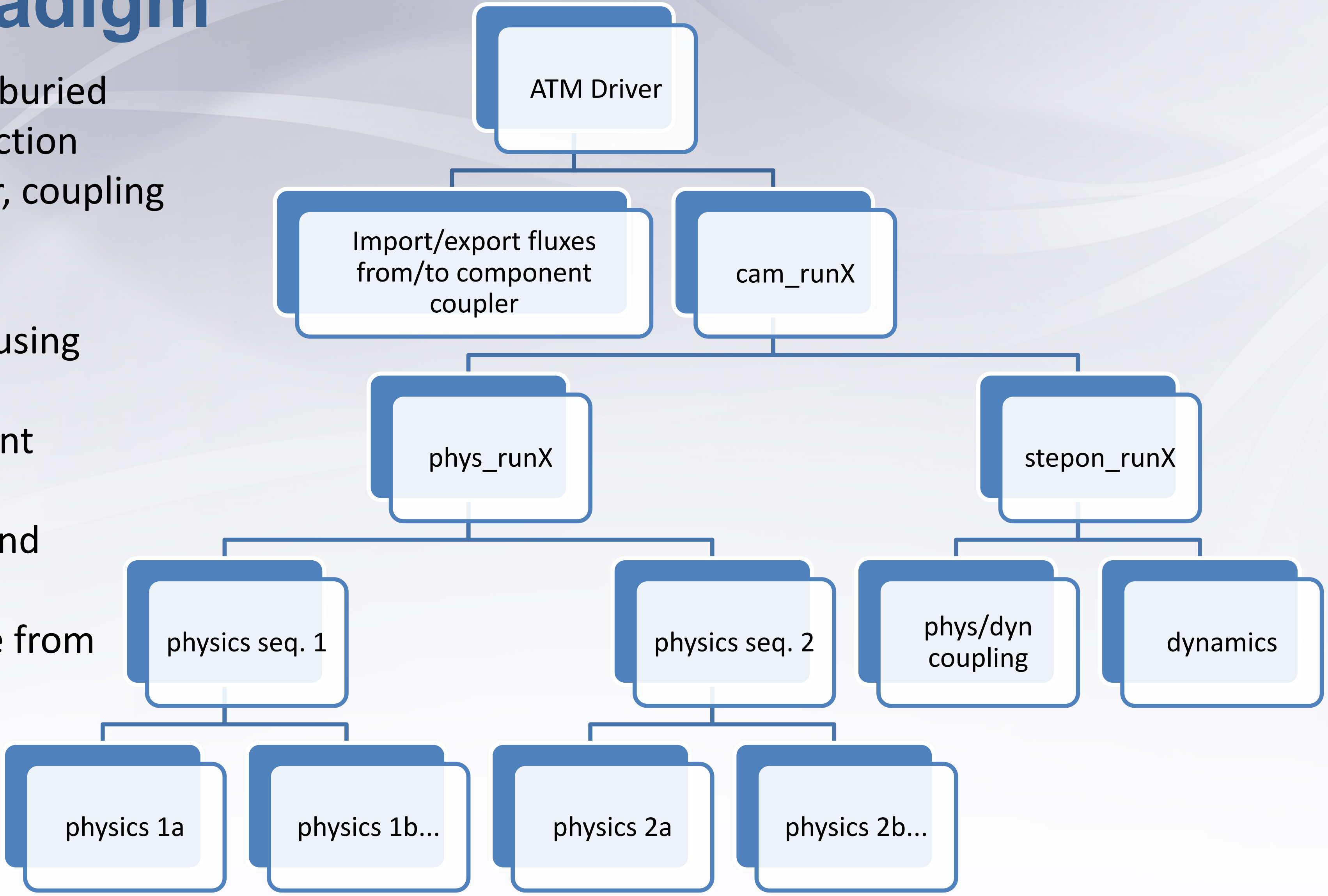
- Controls the coupling of atmospheric processes.
- Controls the passage of information between atmospheric processes.
- Controls the import/export of data from the atmosphere to the other model components.
- Interfaces with the input/output routines.



Current E3SM paradigm

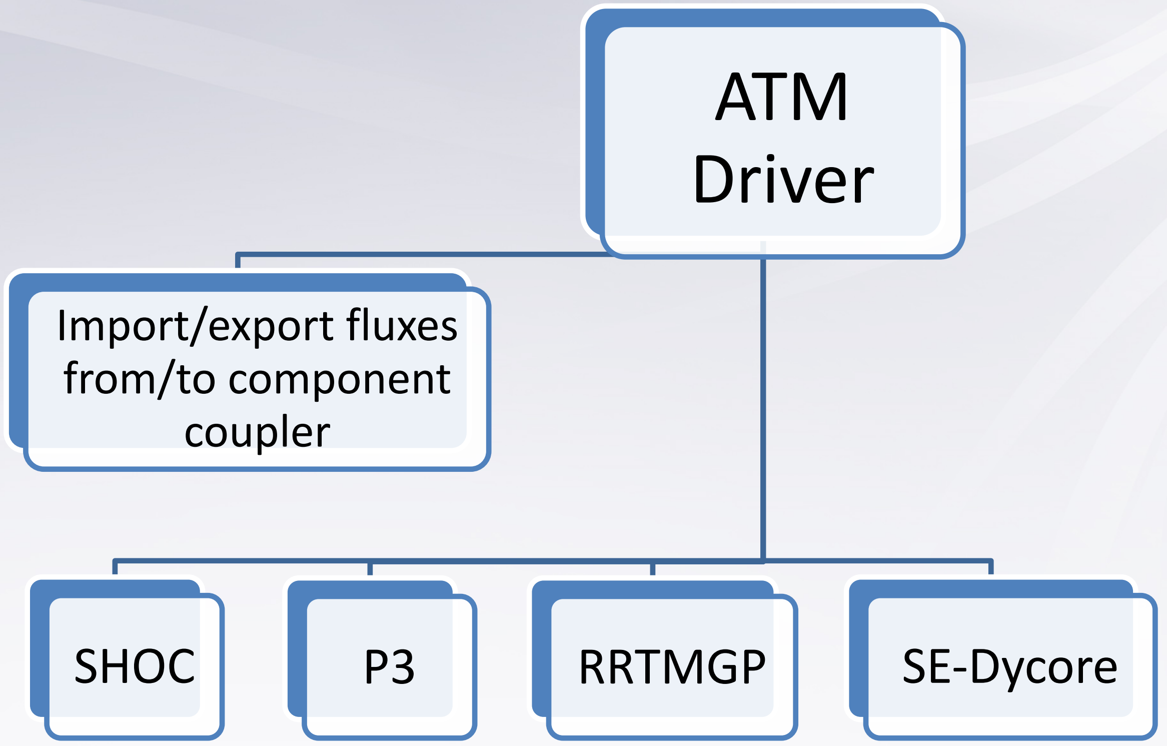
- Actual atmospheric processes are buried beneath multiples layers of abstraction
  - makes changing process order, coupling approach, or adding new parameterizations difficult
  - makes the run sequence confusing

- Different processes require different information, limiting code reuse:
  - Dynamics needs both states and tendencies from physics.
  - Physics receives only the state from dynamics.
  - Only tendencies are passed between parameterizations.



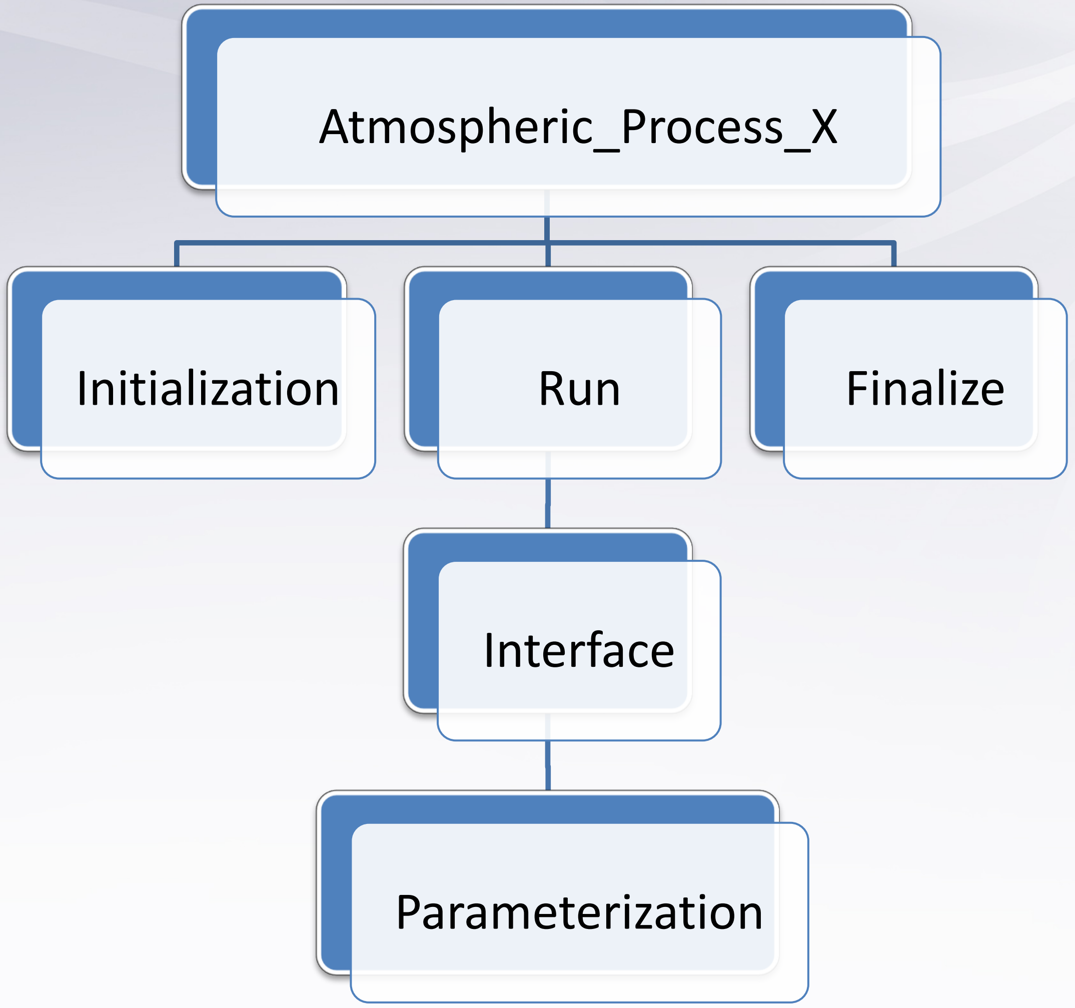
SCREAM Atmospheric Driver

- Uses a generic **atmospheric process class** for both dynamics and physics which is responsible for:
  - The import and export of surface fluxes
  - Interfacing with the set of atmospheric processes
- This simpler paradigm allows for:
  - Straightforward changes to process order
  - Switching between parallel & sequential splitting
  - Easy addition of new parameterizations
- Enables consistent passage of information between processes:
  - Only the model state will be passed in and out of atmospheric processes



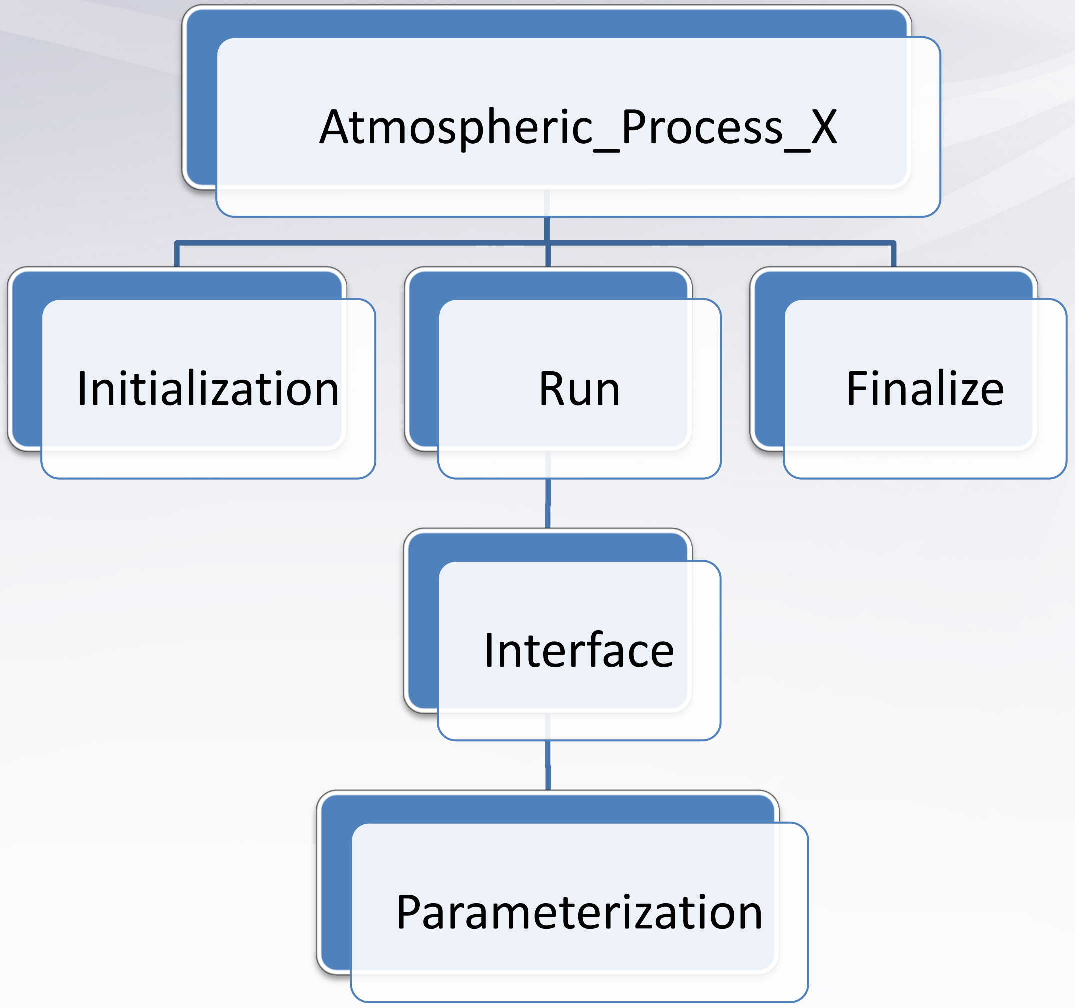
Atmospheric Process Class

- Provides consistent infrastructure for all processes
- Each process has init, run, and finalize methods
- Parameterization portability is enabled by using an ‘interface’ layer to convert input/output between AD- and parameterization-specific data structures



Atmospheric Process Class

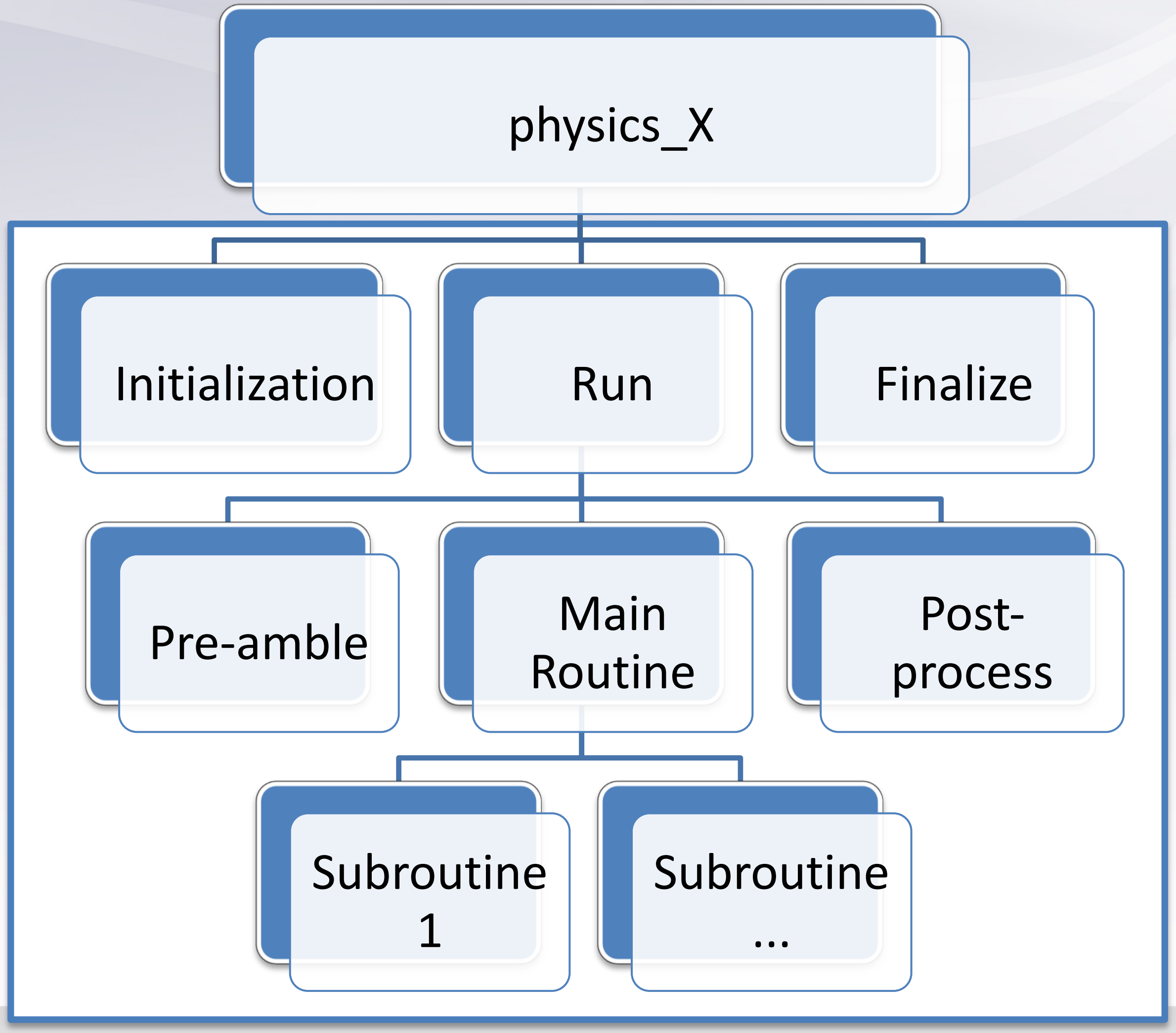
- Provides consistent infrastructure for all processes
- Each process has init, run, and finalize methods
- Parameterization portability is enabled by using an ‘interface’ layer to convert input/output between AD- and parameterization-specific data structures



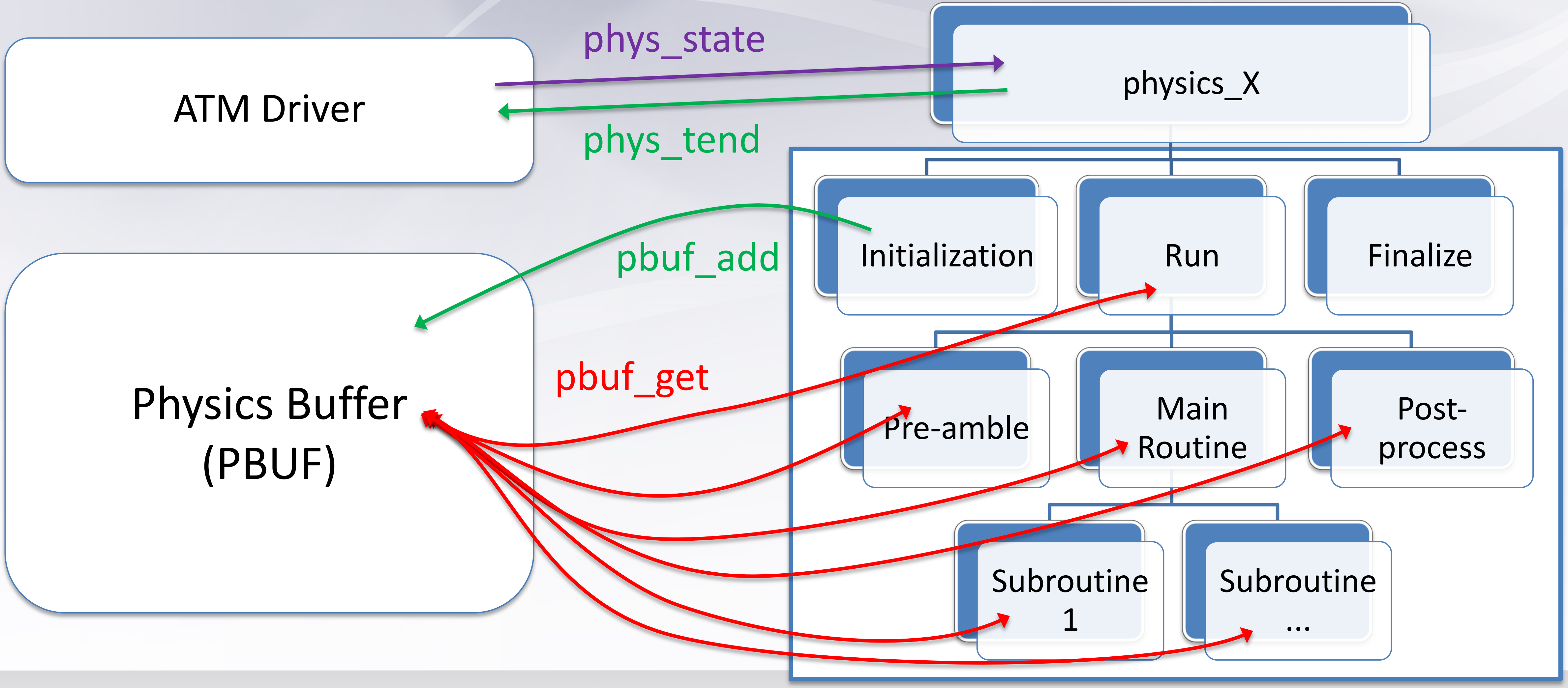
How data is passed around currently:

ATM Driver

Physics Buffer (PBUF)



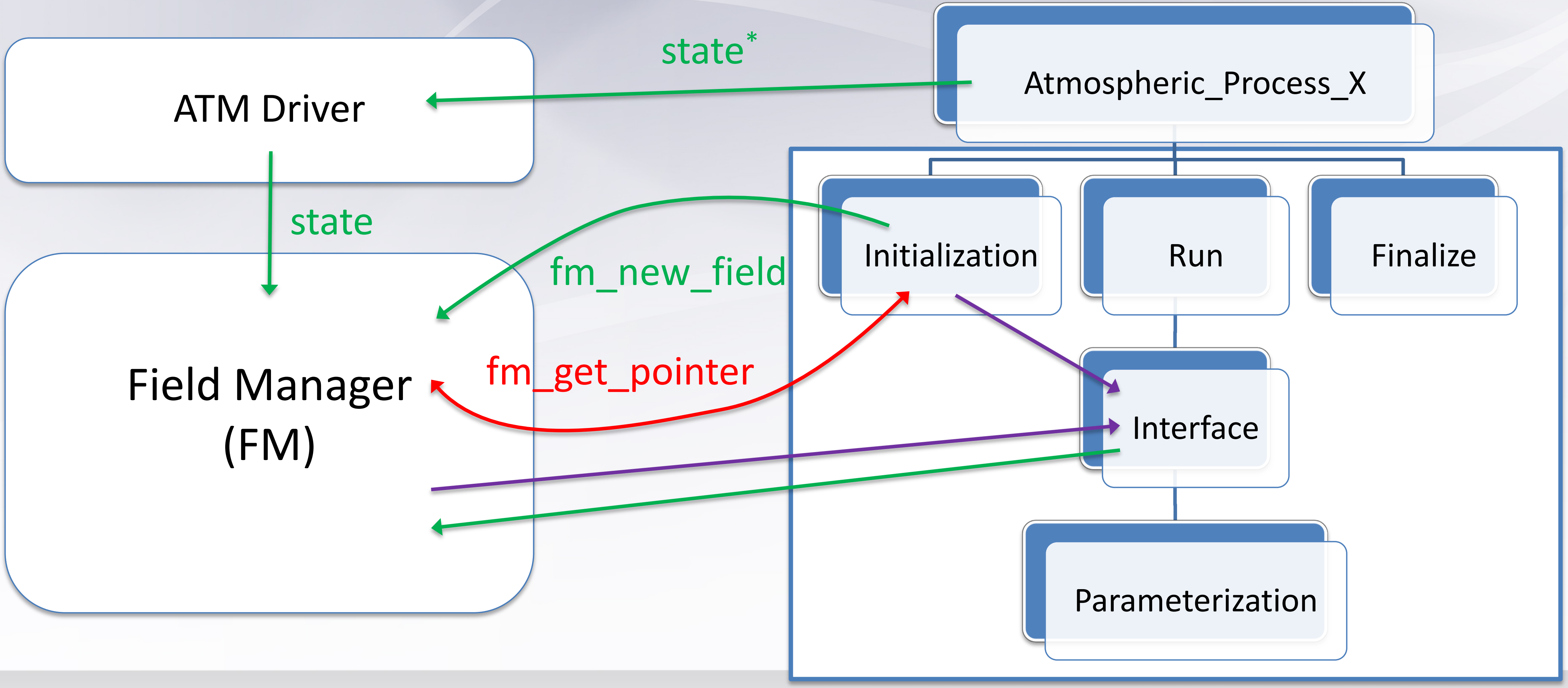
How data is passed around currently



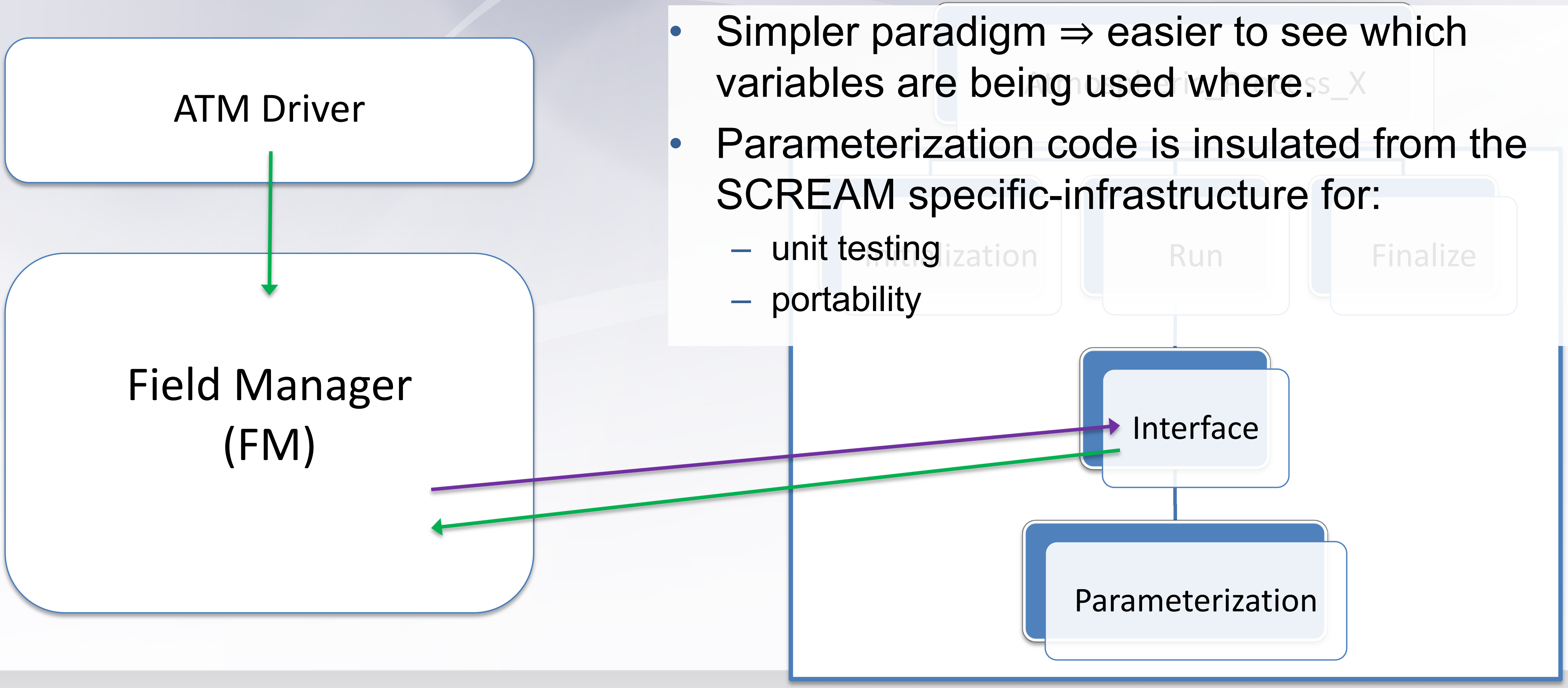
Field Manager (FM)

- Like PBUF, FM will associate variables with pointers to memory.
- FM will handle all AD variables, including prognostic state variables.
  - Only the AD layer will be able to change prognostic state variables.
- FM will only be accessible by initialization and parameterization-interface layers.
  - As a result, all input/output to parameterizations must be passed as input and/or output.
- FM will include new tools to:
  - track where variables are used
  - identify where variables are changed

How data will be passed around in SCREAM FM



How data will be passed around in SCREAM FM



- Simpler paradigm ⇒ easier to see which variables are being used where.
- Parameterization code is insulated from the SCREAM specific-infrastructure for:
  - unit testing
  - portability

Accomplishments and Conclusions

- Written SCREAM-AD design document and will submit for external review shortly.
  - Available on Confluence: https://acme-climate.atlassian.net/wiki/spaces/NGDNA/pages/907378934/SCREAM-AD+Design+Docs
- Ongoing C++ development of SCREAM-AD code.
- The SCREAM-AD maintains the good properties of E3SM's driver logic but simplifies and improves things where possible.
- Our **atmospheric process** class streamlines the interface between the atmosphere model driver and the individual processes.
- A new **field manager class** improves on the current physics buffer structure by
  - simplifying the interface between processes and variables.
  - Insulating parameterization code from model infrastructure, facilitating unit tests and portability.