A step towards SIMA

Implementation of the CCPP in CAM

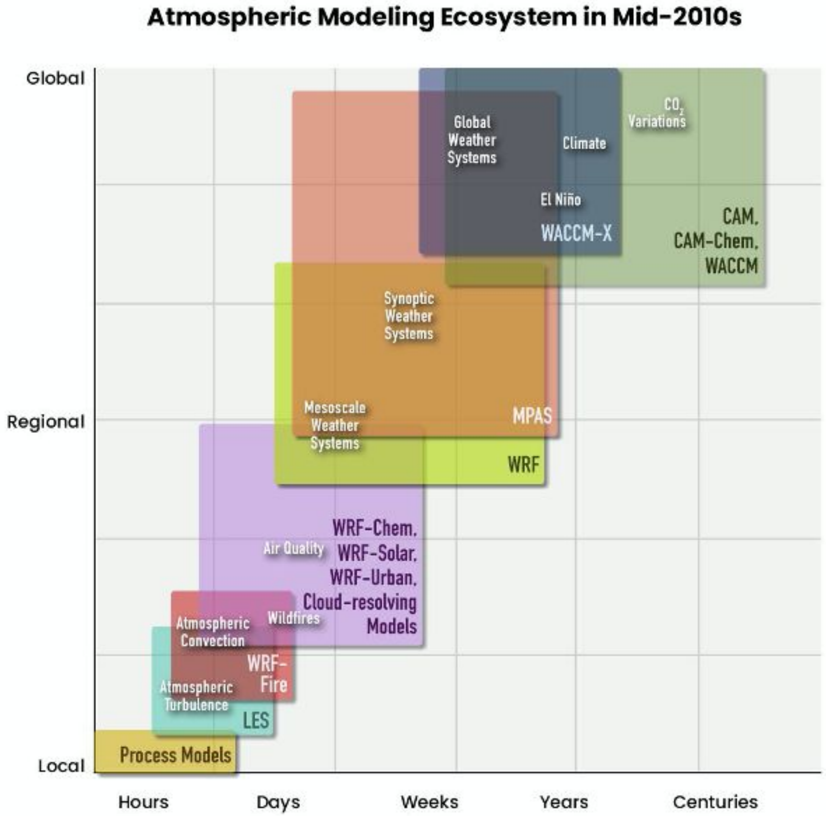
Jesse Nusbaumer,

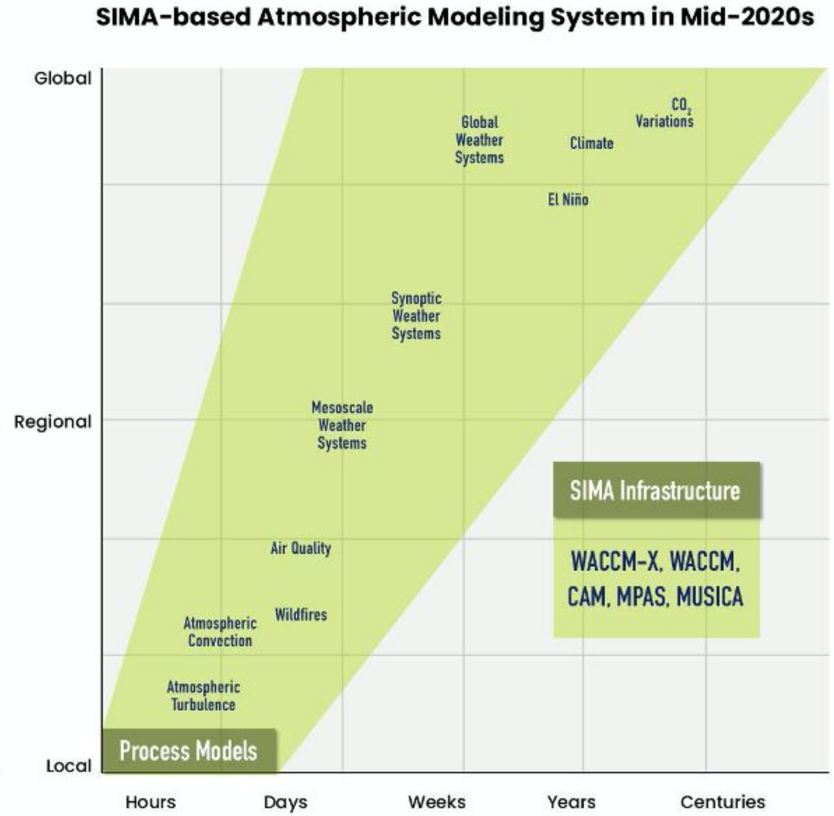
Software Engineer, NCAR CGD-AMP

Jan 30th , 2023

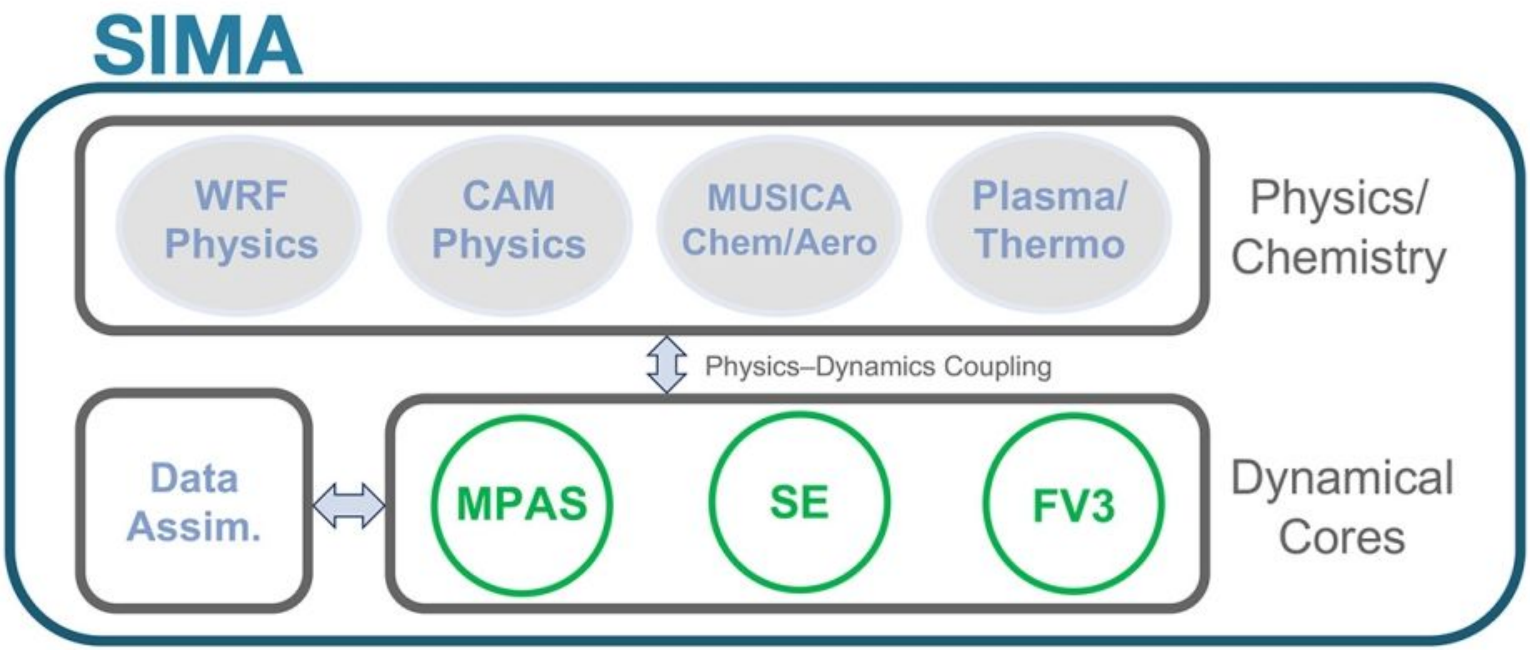
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH · NCAR UCAR · UNIVERSITY CORPORATION FOR ATMOSPHERIC RESEARCH

System for Integrated Modeling of the Atmosphere (SIMA)

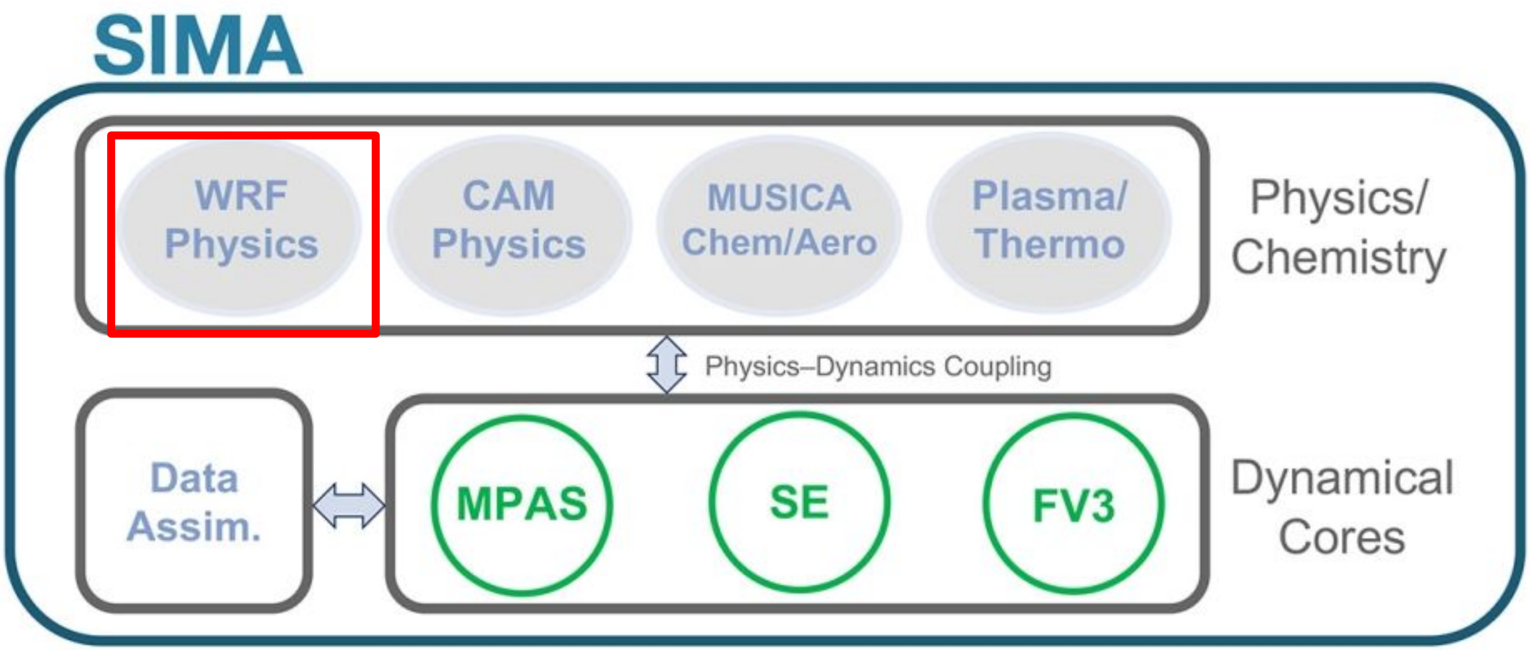




SIMA is a unified community atmospheric modeling framework, for use in an Earth System Model (ESM). SIMA enables diverse configurations of an atmosphere model inside of an ESM for applications spanning minutes to centuries and cloud to global scales, including atmospheric forecasts and projections of the atmospheric state and composition from the surface into the thermosphere.



All of this is already in CAM in some form…



All of this is already in CAM in some form…*except* WRF Physics. How can we bring in entirely new sets of physics into what will eventually be SIMA?

Current CAM physics

```
nusbaume@cheyenne3:~/CESM/escomp_cam/src/physics/cam> ls
aer_rad_props.F90          co2_cycle.F90            gw_rdg.F90                 phys_gmean.F90           subcol_pack_mod.F90.in
aoa_tracers.F90            co2_data_flux.F90        gw_utils.F90               phys_grid_ctem.F90       subcol_SILHS.F90
beljaars_drag_cam.F90      const_init.F90           hb_diff.F90                phys_grid.F90            subcol_tstcp.F90
beljaars_drag.F90          constituent_burden.F90   hetfrz_classnuc_cam.F90    physics_buffer.F90.in    subcol_utils.F90.in
boundarydata.F90           constituents.F90         hetfrz_classnuc.F90        physics_types.F90        tidal_diag.F90
cam3_aero_data.F90         convect_deep.F90         hk_conv.F90                physpkg.F90              tracers.F90
cam3_ozone_data.F90        convect_shallow.F90      iondrag.F90                phys_prop.F90            tracers_suite.F90
cam_diagnostics.F90        conv_water.F90           iop_forcing.F90            pkg_cldoptics.F90        trb_mtn_stress_cam.F90
cam_snapshot.F90           cospsimulator_intr.F90   lunar_tides.F90            pkg_cld_sediment.F90     trb_mtn_stress.F90
carma_flags_mod.F90        cpslec.F90               macrop_driver.F90          polar_avg.F90            tropopause.F90
carma_intr.F90             dadadj_cam.F90           microp_aero.F90            ppgrid.F90               unicon_cam.F90
carma_model_flags_mod.F90  dadadj.F90               microp_driver.F90          qbo.F90                  unicon.F90
check_energy.F90           diffusion_solver.F90     micro_pumas_cam.F90        qneg_module.F90          unicon_utils.F90
chem_surfvals.F90          eddy_diff_cam.F90        modal_aer_opt.F90          rad_constituents.F90     uwshcu.F90
cldfrc2m.F90               eddy_diff.F90            molec_diff.F90             radheat.F90              vdiff_lu_solver.F90
cldwat2m_macro.F90         flux_avg.F90             ndrop_bam.F90              radiation_data.F90       vertical_diffusion.F90
cldwat.F90                 geopotential.F90         ndrop.F90                  rayleigh_friction.F90    waccmx_phys_intr.F90
cloud_cover_diags.F90      ghg_data.F90             nucleate_ice_cam.F90       ref_pres.F90             wv_sat_methods.F90
cloud_diagnostics.F90      gw_common.F90            nucleate_ice.F90           restart_physics.F90      wv_saturation.F90
cloud_fraction.F90         gw_convect.F90           nudging.F90                rk_stratiform.F90        zm_conv.F90
clubb_intr.F90             gw_diffusion.F90         pbl_utils.F90              spcam_drivers.F90        zm_conv_intr.F90
clubb_mf.F90               gw_drag.F90              phys_control.F90           ssatcontrail.F90         zm_microphysics.F90
CMakeLists.txt             gw_front.F90             phys_debug.F90             sslt_rebin.F90
cmparray_mod.F90           gw_oro.F90               phys_debug_util.F90        subcol.F90
nusbaume@cheyenne3:~/CESM/escomp_cam/src/physics/cam>
```

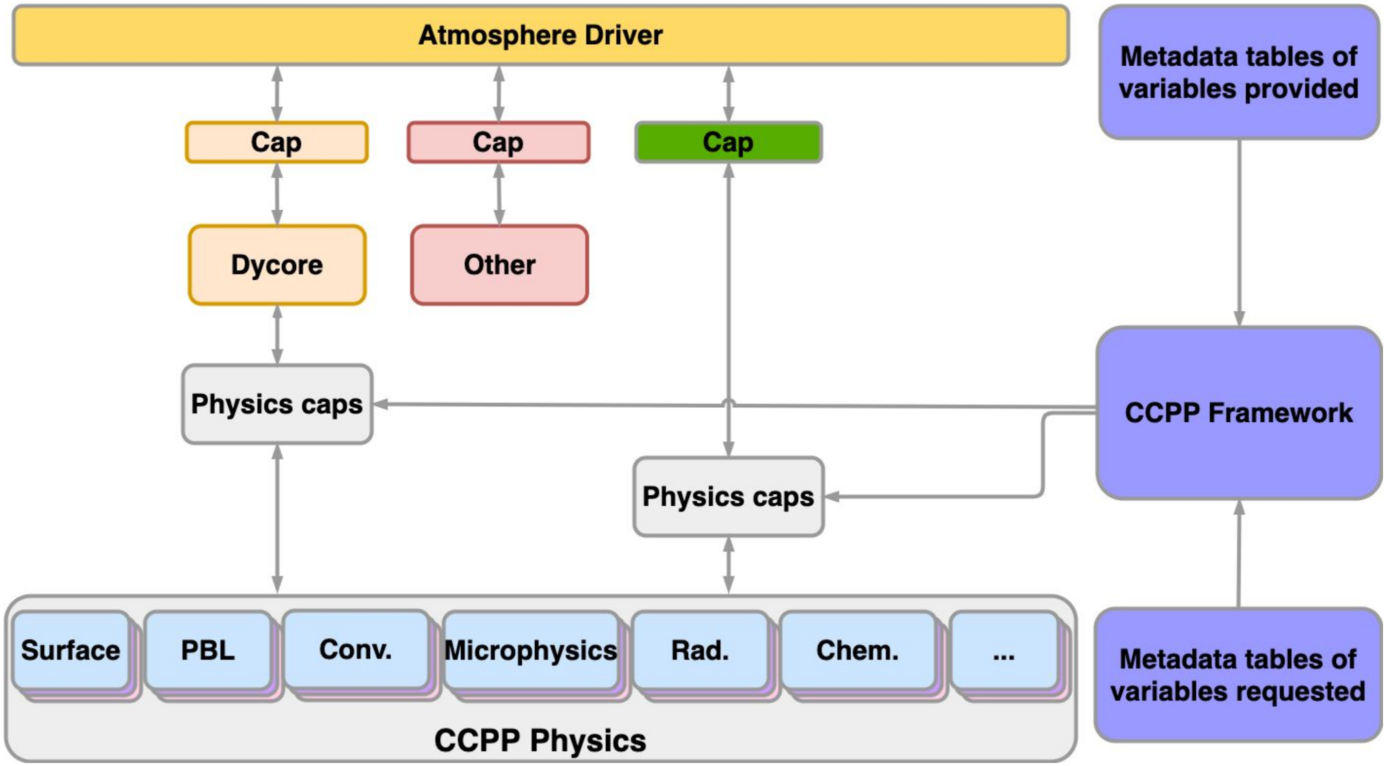
Terminal Window showing all of the “CAM” physics source code

Current CAM physics

```
nusbaume@cheyenne3:~/CESM/escomp_cam/src/physics/cam> ls
aer_rad_props.F90         co2_cycle.F90             gw_rdg.F90                  phys_gmean.F90            subcol_pack_mod.F90.in
aoa_tracers.F90           co2_data_flux.F90         gw_utils.F90                phys_grid_ctem.F90        subcol_SILHS.F90
beljaars_drag_cam.F90     const_init.F90            hb_diff.F90                 phys_grid.F90             subcol_tstcp.F90
beljaars_drag.F90         constituent_burden.F90    hetfrz_classnuc_cam.F90     physics_buffer.F90.in     subcol_utils.F90.in
boundarydata.F90          constituents.F90          hetfrz_classnuc.F90         physics_types.F90         tidal_diag.F90
cam3_aero_data.F90        convect_deep.F90          hk_conv.F90                 physpkg.F90               tracers.F90
cam3_ozone_data.F90       convect_shallow.F90       iondrag.F90                 phys_prop.F90             tracers_suite.F90
cam_diagnostics.F90       conv_water.F90            iop_forcing.F90             pkg_cldoptics.F90         trb_mtn_stress_cam.F90
cam_snapshot.F90          cospsimulator_intr.F90    lunar_tides.F90             pkg_cld_sediment.F90      trb_mtn_stress.F90
carma_flags_mod.F90       cpslec.F90                macrop_driver.F90           polar_avg.F90             tropopause.F90
carma_intr.F90            dadadj_cam.F90            microp_aero.F90             ppgrid.F90                unicon_cam.F90
carma_model_flags_mod.F90 dadadj.F90                microp_driver.F90           qbo.F90                   unicon.F90
check_energy.F90          diffusion_solver.F90      micro_pumas_cam.F90         qneg_module.F90           unicon_utils.F90
chem_surfvals.F90         eddy_diff_cam.F90         modal_aer_opt.F90           rad_constituents.F90      uwshcu.F90
cldfrc2m.F90              eddy_diff.F90             molec_diff.F90              radheat.F90               vdiff_lu_solver.F90
cldwat2m_macro.F90        flux_avg.F90              ndrop_bam.F90               radiation_data.F90        vertical_diffusion.F90
cldwat.F90                geopotential.F90          ndrop.F90                   rayleigh_friction.F90     waccmx_phys_intr.F90
cloud_cover_diags.F90     ghg_data.F90              nucleate_ice_cam.F90        ref_pres.F90              wv_sat_methods.F90
cloud_diagnostics.F90     gw_common.F90             nucleate_ice.F90            restart_physics.F90       wv_saturation.F90
cloud_fraction.F90        gw_convect.F90            nudging.F90                 rk_stratiform.F90         zm_conv.F90
clubb_intr.F90            gw_diffusion.F90          pbl_utils.F90               spcam_drivers.F90         zm_conv_intr.F90
clubb_mf.F90              gw_drag.F90               phys_control.F90            ssatcontrail.F90          zm_microphysics.F90
CMakeLists.txt            gw_front.F90              phys_debug.F90              sslt_rebin.F90
cmparray_mod.F90          gw_oro.F90                phys_debug_util.F90         subcol.F90
nusbaume@cheyenne3:~/CESM/escomp_cam/src/physics/cam>
```

Terminal Window showing all of the “CAM” physics source code, with all files highlighted in green just CAM interface routines.

Common Community Physics Package (CCPP)



The CCPP is a software framework that automatically generates the Fortran interface (cap) layer for a physics parameterization (scheme).

CCPP Suite Definition File

The list and order of physics schemes is controlled by a Suite Definition File (SDF), which allows for much easier re-ordering of physics routines, and removes the need to have a “physpkg.F90” source file.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<suite name="held_suarez_1994" version="1.0">
  <group name="physics">
    <scheme>held_suarez_1994</scheme>
    <scheme>apply_tendency_of_x_wind</scheme>
    <scheme>apply_tendency_of_y_wind</scheme>
    <scheme>apply_heating_rate</scheme>
    <scheme>qneg</scheme>
  </group>
</suite>
```

CCPP Physics Scheme

Metadata (*.meta) file,
which lists metadata for all interface variables

```
[ccpp-arg-table]
  name = apply_tendency_of_x_wind_run
  type = scheme
[ nz ]
  standard_name = vertical_layer_dimension
  long_name = Number of vertical layers
  units = count
  type = integer
  dimensions = ()
  intent = in
[ dudt ]
  standard_name = tendency_of_x_wind
  units = m s-2
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_extent, vertical_layer_dimension)
  intent = in
[ u ]
  standard_name = x_wind
  units = m s-1
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_extent, vertical_layer_dimension)
  intent = inout
  state_variable = True
[ dudt_total ]
  standard_name = tendency_of_x_wind_due_to_model_physics
  units = m s-2
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_extent, vertical_layer_dimension)
  intent = inout
```

Source code (*.F90) file,
which contains the actual parameterization code

```
!> \section arg_table_apply_tendency_of_x_wind_run  Argument Table
! \htmlinclude apply_tendency_of_x_wind_run.html
subroutine apply_tendency_of_x_wind_run(nz, dudt, u, dudt_total, dt, &
                                        errcode, errmsg)

  ! Dummy arguments
  integer,                intent(in)     :: nz              ! Num vertical  layers
  real(kind_phys),        intent(in)     :: dudt(:,:)       ! tendency of x wind
  real(kind_phys),        intent(inout)  :: u(:,:)          ! x wind
  real(kind_phys),        intent(inout)  :: dudt_total(:,:) ! total tendency of x wind
  real(kind_phys),        intent(in)     :: dt              ! physics time step
  integer,                intent(out)    :: errcode
  character(len=512),     intent(out)    :: errmsg

  ! Local variable
  integer :: klev

  errcode = 0
  errmsg = ''

  do klev = 1, nz
    u(:, klev) = u(:, klev) + (dudt(:, klev) * dt)
    dudt_total(:, klev) = dudt_total(:, klev) + dudt(:, klev)
  end do

end subroutine apply_tendency_of_x_wind_run
```

With these two files and a host model metadata file, the model/scheme interface can be auto-generated

CCPP Physics Scheme

Metadata (*.meta) file,
which lists metadata for all interface variables

```
[ccpp-arg-table]
  name = apply_tendency_of_x_wind_run
  type = scheme
[ nz ]
  standard_name = vertical_layer_dimension
  long_name = Number of vertical layers
  units = count
  type = integer
  dimensions = ()
  intent = in
[ dudt ]
  standard_name = tendency_of_x_wind
  units = m s-2
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_exten
  intent = in
[ u ]
  standard_name = x_wind
  units = m s-1
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_extent, vertical_layer_dimension)
  intent = inout
  state_variable = True
[ dudt_total ]
  standard_name = tendency_of_x_wind_due_to_model_physics
  units = m s-2
  type = real | kind = kind_phys
  dimensions = (horizontal_loop_extent, vertical_layer_dimension)
  intent = inout
```

Source code (*.F90) file,
which contains the actual parameterization code

```
!> \section arg_table_apply_tendency_of_x_wind_run  Argument Table
! \htmlinclude apply_tendency_of_x_wind_run.html
subroutine apply_tendency_of_x_wind_run(nz, dudt, u, dudt_total, dt, &
                                        errcode, errmsg)

  ! Dummy arguments
  integer,             intent(in)     :: nz              ! Num vertical  layers
  real(kind_phys),     intent(in)     :: dudt(:,:)       ! tendency of x wind
  real(kind_phys),     intent(inout)  :: u(:,:)          ! x wind
  real(kind_phys),     intent(inout)  :: dudt_total(:,:) ! total tendency of x wind
  real(kind_phys),     intent(in)     :: dt              ! physics time step
  integer,             intent(out)    :: errcode
                                         errmsg

  errcode = 0
  errmsg = ''

  do klev = 1, nz
    u(:, klev) = u(:, klev) + (dudt(:, klev) * dt)
    dudt_total(:, klev) = dudt_total(:, klev) + dudt(:, klev)
  end do

end subroutine apply_tendency_of_x_wind_run
```

All schemes are split into a model init, timestep init, run, timestep final, and model final phase.

With these two files and a host model metadata file, the model/scheme interface can be auto-generated

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

1. Save a snapshot of the model state before and after the CAM scheme

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

1. Save a snapshot of the model state before and after the CAM scheme

2. Create a metadata file for that scheme, and pull out the source code to be CCPP-compliant

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

1. Save a snapshot of the model state before and after the CAM scheme

2. Create a metadata file for that scheme, and pull out the source code to be CCPP-compliant

3. Add at least the “run” phase of the new CCPP scheme back into CAM, and check that it is bit-for-bit

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

1. Save a snapshot of the model state before and after the CAM scheme

2. Create a metadata file for that scheme, and pull out the source code to be CCPP-compliant

3. Add at least the "run" phase of the new CCPP scheme back into CAM, and check that it is bit-for-bit

4. Add the full CCPP scheme into CAMDEN (described in next slide)

CCPP Implementation plan

For each physics parameterization/scheme in CAM, the SEs will:

1. Save a snapshot of the model state before and after the CAM scheme

2. Create a metadata file for that scheme, and pull out the source code to be CCPP-compliant

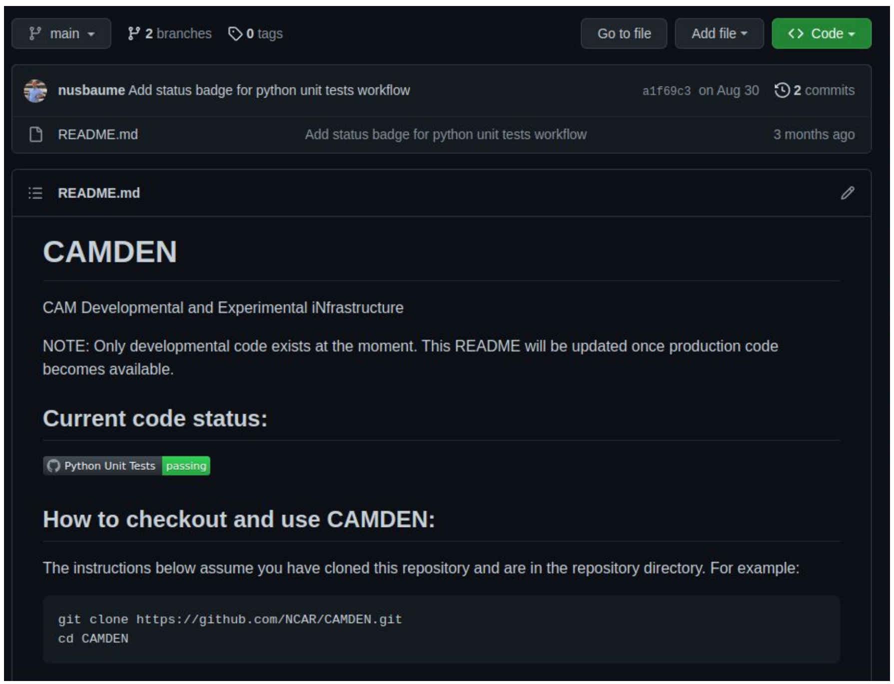
3. Add at least the "run" phase of the new CCPP scheme back into CAM, and check that it is bit-for-bit

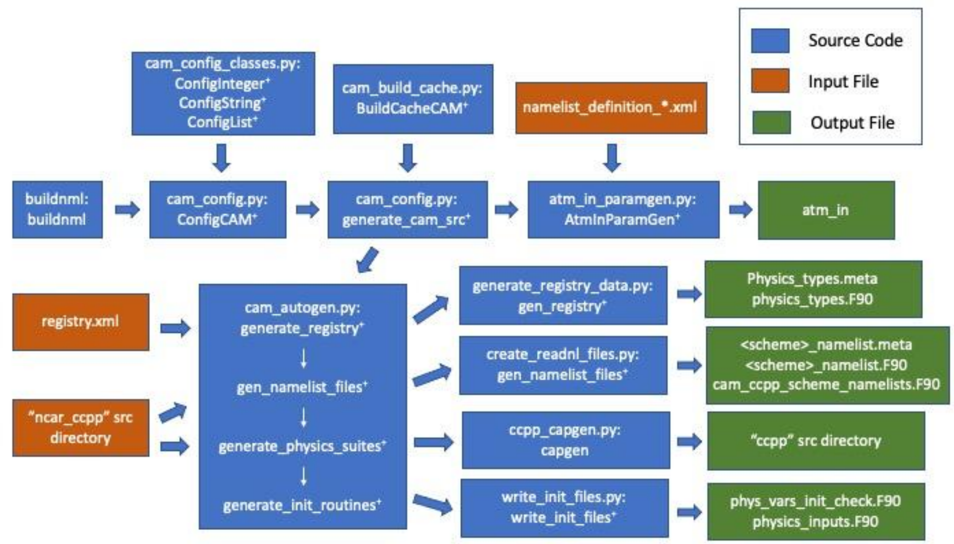
4. Add the full CCPP scheme into CAMDEN (described in next slide)

5. Test the full scheme in CAMDEN using the snapshots, and ensure that the answers are the same.

CAMDEN

CAMDEN is a new model infrastructure for CAM, which should become publicly accessible sometime later this year (although it will have limited science capabilities at first).





People and Time

Folks who will be working on this transition include myself and:

- Courtney Peverley <- Soon to be AMP's CCPP framework expert
- Cheryl Craig
- John Truesdale
- Kate Thayer-Calder
- Peter Lauritzen
- AMP Scientists

The current hope is to have the CAM7 (cam_dev) physics schemes ported to CCPP by this time next year, but will depend on a myriad of factors.

Questions

Thanks for listening!

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
NCAR
UCAR
UNIVERSITY CORPORATION FOR ATMOSPHERIC RESEARCH