Reworking CTSM's Water Flux Calculations

Bill Sacks
CESM Software Engineering Group (CSEG)

with many contributions from Mat Rothstein (CSEG)

and also: Mike Barlage, Martyn Clark, Dave Lawrence, David Noone, Sean Swenson, Mariana Vertenstein

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

NSF

Outline

- What is the problem?
- Why is this a problem?
- What are we doing?
- Two metaphors that inspire me

What is the problem? (1)

```
subroutine SurfaceRunoff (...)
  ! Inputs: [cut for brevity]
  !
  ! Input/outputs:
  ! - h2osoi_liq: liquid water (mm)
  ! - qflx_top_soil: net water input into soil from top (mm/s)
  !
  ! Outputs:
  ! - qflx_surf: surface runoff (mm/s)
  ! - fcov: fractional impermeable area
  ! - fsat: fractional area with water table at surface
  ! - fracice: fractional impermeability
  ! - icefrac: fraction of ice
  ! - max_infil: maximum infiltration capacity in VIC (mm)
  ! - i_0: column average soil moisture in top VIC layers (mm)
```

What is the problem? (2)

```
do j = 1,nlevsoi
   do fc = 1, num_hydrologyc
      c = filter_hydrologyc(fc)
      ! --- Set icefrac and fracice ---
   end do
end do
```

```
do fc = 1, num_hydrologyc
   c = filter_hydrologyc(fc)
   if (use_vichydro) then
      ! --- Set a bunch of auxiliary VIC variables ---

      fsat(c) = ...
   else
      fsat(c) = ...
   end if
```

What is the problem? (3)

```
    if ( frost_table(c) > zwt(c)) then
       if (use_vichydro) then
          fsat(c) = ...
       else
          fsat(c) = ...
       end if
    else
       if ( frost_table(c) > zwt_perched(c)) then
          fsat(c) = ...
       end if
    end if


 end do
```

What is the problem? (4)

```
do fc = 1, num_hydrologyc
   c = filter_hydrologyc(fc)
   qflx_surf(c) = fsat(c) * qflx_top_soil(c)
end do

do fc = 1, num_urbanc
   c = filter_urbanc(fc)
   ! --- For urban: set qflx_surf   ---
   ! --- and update h2osoi_liq(c,1) ---
  end do

  do fc = 1, num_hydrologyc
     c = filter_hydrologyc(fc)
     ! --- Add some terms to qflx_top_soil ---
  end do

end subroutine SurfaceRunoff
```

What is the problem? (Full routine)

```
!-----------------------------------------------------------------------
subroutine SurfaceRunoff (bounds, num_hydrologyc, filter_hydrologyc, &
     num_urbanc, filter_urbanc, soilhydrology_inst, soilstate_inst,
waterflux_inst, &
     waterstate_inst)
  !
  ! !DESCRIPTION:
  ! Calculate surface runoff
  !
  ! !USES:
  use clm_varcon      , only : denice, denh2o, wimp, pondmx_urban
  use column_varcon   , only : icol_roof, icol_sunwall, icol_shadewall
  use column_varcon   , only : icol_road_imperv, icol_road_perv
  use clm_varpar      , only : nlevsoi, maxpatch_pft
  use clm_time_manager, only : get_step_size
  use clm_varpar      , only : nlayer, nlayert
  use abortutils      , only : endrun
  !
  ! !ARGUMENTS:
  type(bounds_type)        , intent(in)    :: bounds
  integer                  , intent(in)    :: num_hydrologyc       ! number of
column soil points in column filter
  integer                  , intent(in)    :: filter_hydrologyc(:) ! column
filter for soil points
  integer                  , intent(in)    :: num_urbanc           ! number of
column urban points in column filter
  integer                  , intent(in)    :: filter_urbanc(:)     ! column
filter for urban points
  type(soilhydrology_type) , intent(inout) :: soilhydrology_inst
  type(soilstate_type)     , intent(in)    :: soilstate_inst
  type(waterflux_type)     , intent(inout) :: waterflux_inst
  type(waterstate_type)    , intent(inout) :: waterstate_inst
  !
  ! !LOCAL VARIABLES:
  integer  :: c,j,fc,g,l,i                                !indices
  real(r8) :: dtime                                       !land model time step
(sec)
  real(r8) :: xs(bounds%begc:bounds%endc)                 !excess soil water
above urban ponding limit
  real(r8) :: vol_ice(bounds%begc:bounds%endc,1:nlevsoi) !partial volume of ice
lens in layer
  real(r8) :: fff(bounds%begc:bounds%endc)                !decay factor (m-1)
  real(r8) :: s1                                          !variable to calculate
qinmax
  real(r8) :: su                                          !variable to calculate
qinmax
  real(r8) :: v                                           !variable to calculate
qinmax
  real(r8) :: qinmax                                      !maximum infiltration
capacity (mm/s)
  real(r8) :: A(bounds%begc:bounds%endc)                  !fraction of the
saturated area
  real(r8) :: ex(bounds%begc:bounds%endc)                 !temporary variable
(exponent)
  real(r8) :: top_moist(bounds%begc:bounds%endc)          !temporary, soil
moisture in top VIC layers
  real(r8) :: top_max_moist(bounds%begc:bounds%endc)      !temporary, maximum
soil moisture in top VIC layers
  real(r8) :: top_ice(bounds%begc:bounds%endc)            !temporary, ice len in
top VIC layers
  character(len=32) :: subname = 'SurfaceRunoff'          !subroutine name
  !-----------------------------------------------------------------------

  associate(                                                &
       snl             =>    col%snl                            , & !
Input:  [integer  (:)   ]  minus number of snow layers
       dz              =>    col%dz                             , & !
Input:  [real(r8) (:,:) ]  layer depth (m)

       sucsat          =>    soilstate_inst%sucsat_col          , & !
Input:  [real(r8) (:,:) ]  minimum soil suction (mm)
       watsat          =>    soilstate_inst%watsat_col          , & !
Input:  [real(r8) (:,:) ]  volumetric soil water at saturation (porosity)
       wtfact          =>    soilstate_inst%wtfact_col          , & !
Input:  [real(r8) (:)   ]  maximum saturated fraction for a gridcell
       hksat           =>    soilstate_inst%hksat_col           , & !
Input:  [real(r8) (:,:) ]  hydraulic conductivity at saturation (mm H2O /s)
       bsw             =>    soilstate_inst%bsw_col             , & !
Input:  [real(r8) (:,:) ]  Clapp and Hornberger "b"

       h2osoi_ice      =>    waterstate_inst%h2osoi_ice_col     , & !
Input:  [real(r8) (:,:) ]  ice lens (kg/m2)
       h2osoi_liq      =>    waterstate_inst%h2osoi_liq_col     , & ! Output:
[real(r8) (:,:) ]  liquid water (kg/m2)

       qflx_snow_h2osfc =>   waterflux_inst%qflx_snow_h2osfc_col , & !
Input:  [real(r8) (:)   ]  snow falling on surface water (mm/s)
       qflx_floodc     =>    waterflux_inst%qflx_floodc_col     , & !
Input:  [real(r8) (:)   ]  column flux of flood water from RTM
       qflx_evap_grnd  =>    waterflux_inst%qflx_evap_grnd_col  , & !
Input:  [real(r8) (:)   ]  ground surface evaporation rate (mm H2O/s) [+]
       qflx_top_soil   =>    waterflux_inst%qflx_top_soil_col   , & ! Output:
[real(r8) (:)   ]  net water input into soil from top (mm/s)
       qflx_surf       =>    waterflux_inst%qflx_surf_col       , & ! Output:
[real(r8) (:)   ]  surface runoff (mm H2O /s)

       zwt             =>    soilhydrology_inst%zwt_col         , & !
Input:  [real(r8) (:)   ]  water table depth (m)
       max_moist       =>    soilhydrology_inst%max_moist_col   , & !
Input:  [real(r8) (:,:) ]  maximum soil moisture (ice + liq, mm)
       frost_table     =>    soilhydrology_inst%frost_table_col , & !
Input:  [real(r8) (:)   ]  frost table depth (m)
       zwt_perched     =>    soilhydrology_inst%zwt_perched_col , & !
Input:  [real(r8) (:)   ]  perched water table depth (m)
       b_infil         =>    soilhydrology_inst%b_infil_col     , & !
Input:  [real(r8) (:)   ]  VIC b infiltration parameter
       moist           =>    soilhydrology_inst%moist_col       , & !
Input:  [real(r8) (:,:) ]  soil moisture in each VIC layers (liq, mm)
       hkdepth         =>    soilhydrology_inst%hkdepth_col     , & !
Input:  [real(r8) (:)   ]  decay factor (m)
       origflag        =>    soilhydrology_inst%origflag        , & !
Input:  logical
       fcov            =>    soilhydrology_inst%fcov_col        , & ! Output:
[real(r8) (:)   ]  fractional impermeable area
       fsat            =>    soilhydrology_inst%fsat_col        , & ! Output:
[real(r8) (:)   ]  fractional area with water table at surface
       fracice         =>    soilhydrology_inst%fracice_col     , & ! Output:
[real(r8) (:,:) ]  fractional impermeability (-)
       icefrac         =>    soilhydrology_inst%icefrac_col     , & ! Output:
[real(r8) (:,:) ]
       ice             =>    soilhydrology_inst%ice_col         , & ! Output:
[real(r8) (:,:) ]  ice len in each VIC layers(ice, mm)
       max_infil       =>    soilhydrology_inst%max_infil_col   , & ! Output:
[real(r8) (:)   ]  maximum infiltration capacity in VIC (mm)
       i_0             =>    soilhydrology_inst%i_0_col         , & ! Output:
[real(r8) (:)   ]  column average soil moisture in top VIC layers (mm)
       )

  ! Get time step

  dtime = get_step_size()

  do j = 1,nlevsoi
     do fc = 1, num_hydrologyc
        c = filter_hydrologyc(fc)

        ! Porosity of soil, partial volume of ice and liquid, fraction of ice in
each layer,
        ! fractional impermeability

        vol_ice(c,j) = min(watsat(c,j), h2osoi_ice(c,j)/(dz(c,j)*denice))
        if (origflag == 1) then
           icefrac(c,j) = min(1._r8,h2osoi_ice(c,j)/(h2osoi_ice(c,j)
+h2osoi_liq(c,j)))
        else
           icefrac(c,j) = min(1._r8,vol_ice(c,j)/watsat(c,j))
        endif

        fracice(c,j) = max(0._r8,exp(-3._r8*(1._r8-icefrac(c,j)))- exp(-3._r8))/
(1.0_r8-exp(-3._r8))
     end do
  end do

  ! Saturated fraction

  do fc = 1, num_hydrologyc
     c = filter_hydrologyc(fc)
     fff(c) = 0.5_r8
     if (use_vichydro) then
        top_moist(c) = 0._r8
        top_ice(c) = 0._r8
        top_max_moist(c) = 0._r8
        do j = 1, nlayer - 1
           top_ice(c) = top_ice(c) + ice(c,j)
           top_moist(c) =  top_moist(c) + moist(c,j) + ice(c,j)
           top_max_moist(c) = top_max_moist(c) + max_moist(c,j)
        end do
        if(top_moist(c)> top_max_moist(c)) top_moist(c)= top_max_moist(c)
        top_ice(c)     = max(0._r8,top_ice(c))
        max_infil(c)   = (1._r8+b_infil(c)) * top_max_moist(c)
        ex(c)          = b_infil(c) / (1._r8 + b_infil(c))
        A(c)           = 1._r8 - (1._r8 - top_moist(c) /
top_max_moist(c))**ex(c)
        i_0(c)         = max_infil(c) * (1._r8 - (1._r8 - A(c))**(1._r8/
b_infil(c)))
        fsat(c)        = A(c)  !for output
     else
        fsat(c) = wtfact(c) * exp(-0.5_r8*fff(c)*zwt(c))
        end if

        ! use perched water table to determine fsat (if present)
        if ( frost_table(c) > zwt(c)) then
           if (use_vichydro) then
              fsat(c) =  A(c)
           else
              fsat(c) = wtfact(c) * exp(-0.5_r8*fff(c)*zwt(c))
           end if
        else
           if ( frost_table(c) > zwt_perched(c)) then
              fsat(c) = wtfact(c) * exp(-0.5_r8*fff(c)*zwt_perched(c))!
*( frost_table(c) - zwt_perched(c))/4.0
           endif
        endif
        if (origflag == 1) then
           if (use_vichydro) then
              call endrun(msg="VICHYDRO is not available for origflag=1"//
errmsg(sourcefile, __LINE__))
           else
              fcov(c) = (1._r8 - fracice(c,1)) * fsat(c) + fracice(c,1)
           end if
        else
           fcov(c) = fsat(c)
        endif
     end do

     do fc = 1, num_hydrologyc
        c = filter_hydrologyc(fc)

        ! assume qinmax large relative to qflx_top_soil in control
        if (origflag == 1) then
           qflx_surf(c) =  fcov(c) * qflx_top_soil(c)
        else
           ! only send fast runoff directly to streams
           qflx_surf(c) =   fsat(c) * qflx_top_soil(c)
        endif
     end do

     ! Determine water in excess of ponding limit for urban roof and impervious
road.
     ! Excess goes to surface runoff. No surface runoff for sunwall and shadewall.

     do fc = 1, num_urbanc
        c = filter_urbanc(fc)
        if (col%itype(c) == icol_roof .or. col%itype(c) == icol_road_imperv) then

           ! If there are snow layers then all qflx_top_soil goes to surface runoff
           if (snl(c) < 0) then
              qflx_surf(c) = max(0._r8,qflx_top_soil(c))
           else
              xs(c) = max(0._r8, &
                   h2osoi_liq(c,1)/dtime + qflx_top_soil(c) - qflx_evap_grnd(c) - &
                   pondmx_urban/dtime)
              if (xs(c) > 0.) then
                 h2osoi_liq(c,1) = pondmx_urban
              else
                 h2osoi_liq(c,1) = max(0._r8,h2osoi_liq(c,1)+ &
                      (qflx_top_soil(c)-qflx_evap_grnd(c))*dtime)
              end if
              qflx_surf(c) = xs(c)
           end if
        else if (col%itype(c) == icol_sunwall .or. col%itype(c) == icol_shadewall)
then
           qflx_surf(c) = 0._r8
        end if
        ! send flood water flux to runoff for all urban columns
        qflx_surf(c) = qflx_surf(c)  + qflx_floodc(c)

     end do

     ! remove stormflow and snow on h2osfc from qflx_top_soil
     do fc = 1, num_hydrologyc
        c = filter_hydrologyc(fc)
        ! add flood water flux to qflx_top_soil
        qflx_top_soil(c) = qflx_top_soil(c) + qflx_snow_h2osfc(c) + qflx_floodc(c)
     end do

     end associate

end subroutine SurfaceRunoff
```

Why is this a problem?
(Right brain)

Why is this a problem?
(Right brain)

Why is this a problem?
(Right brain)

Why is this a problem?
(Right brain)

Why is this a problem? (Left brain)

- Hard to understand and modify the code
- Hard to bring in alternative parameterizations
  - Incorporating Noah-MP parameterizations into CTSM
- Numerics (state updates) entangled with physics: can't implement better numerical solutions
- Hard to implement water tracers / isotopes

Why is this a problem?
Water tracers / isotopes

$$Flux_{tracer} = Flux_{bulk} \cdot \frac{Source_{tracer}}{Source_{bulk}}$$

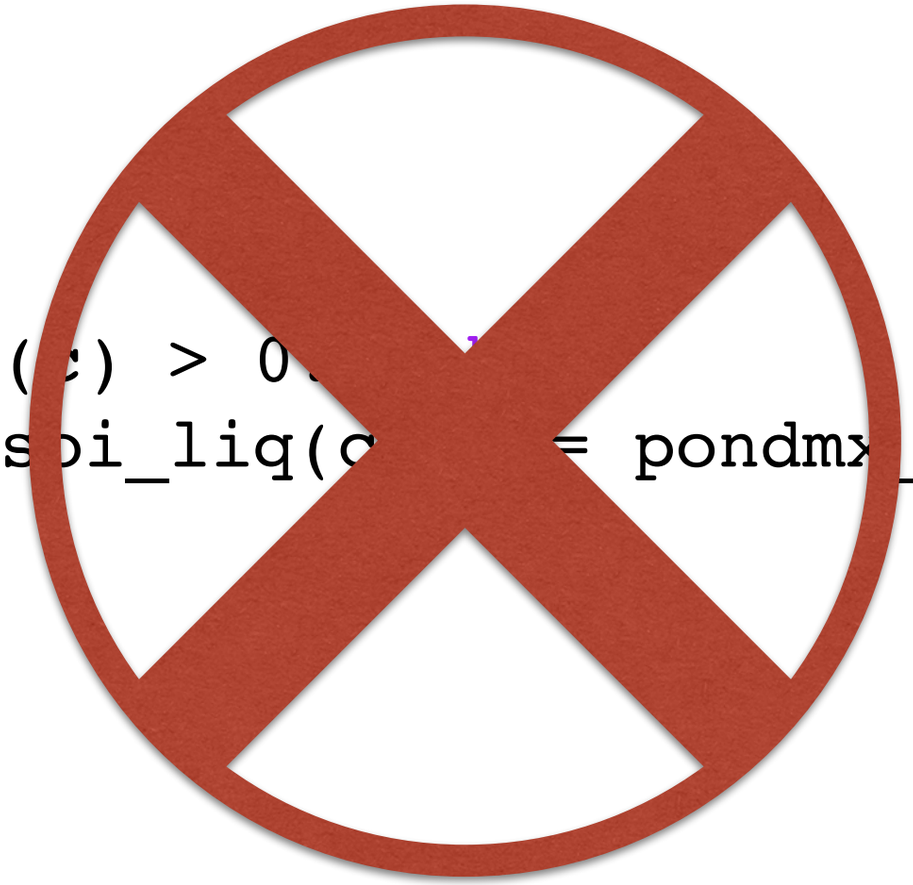
Why is this a problem?
Water tracers / isotopes

$$Flux_{tracer} = Flux_{bulk} \cdot \frac{Source_{tracer}}{Source_{bulk}}$$

```
if (xs(c) > 0.) then
    h2osoi_liq(c,1) = pondmx_urban
```

Why is this a problem?
Water tracers / isotopes

$$Flux_{tracer} = Flux_{bulk} \cdot \frac{Source_{tracer}}{Source_{bulk}}$$



What are we doing? – schematic

Before

some_subroutine
calc. flux 1 … … … update state 1 calc. flux 2 … … … calc. flux 3 … … … update state 2

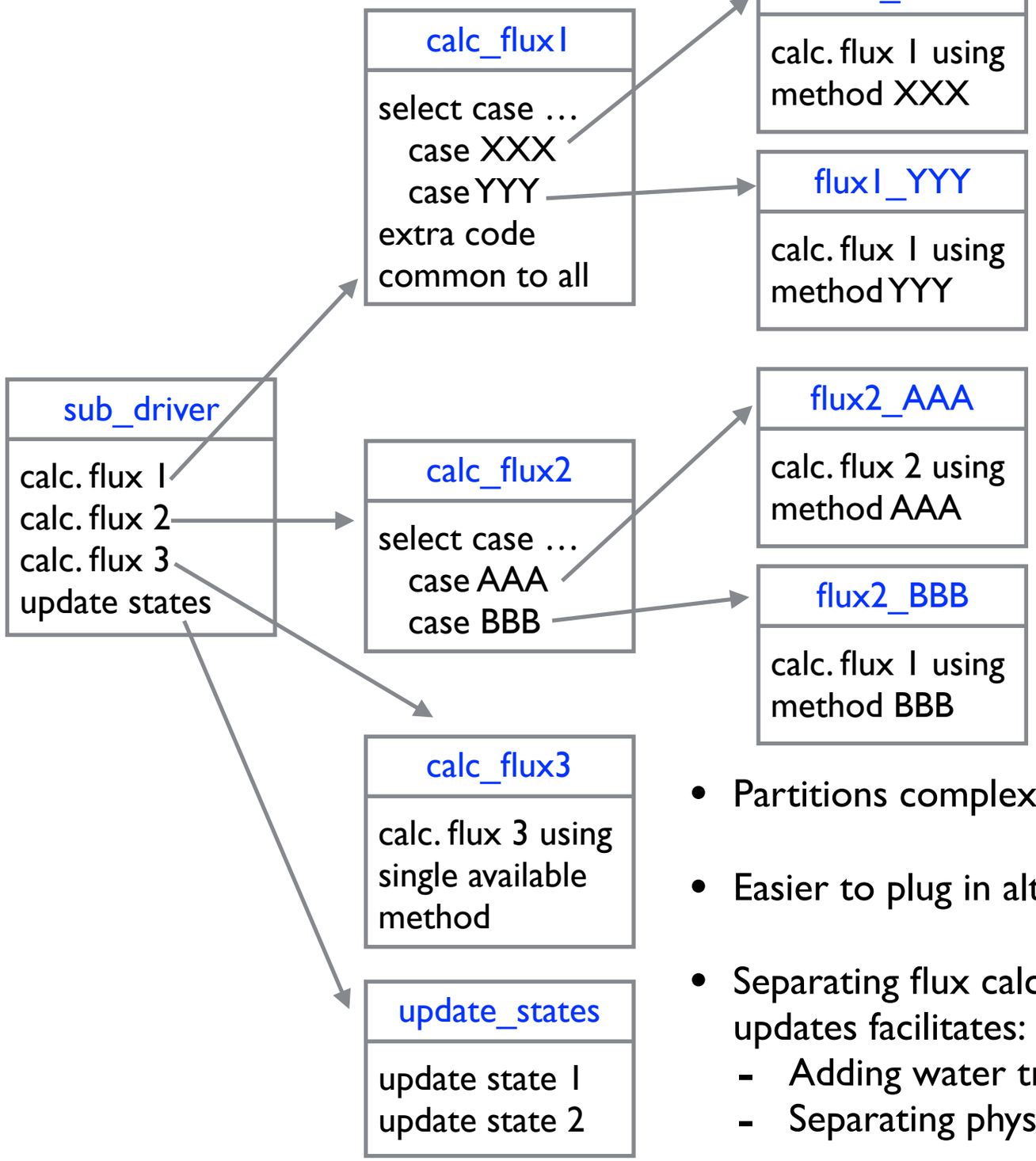
What are we doing? – schematic

Before

some_subroutine

calc. flux 1
…
…
…
update state 1
calc. flux 2
…
…
…
calc. flux 3
…
…
…
update state 2

After



Benefits

- Partitions complexity
- Easier to plug in alternative parameterizations
- Separating flux calculations from state updates facilitates:
  - Adding water tracers / isotopes
  - Separating physics from numerics

What are we doing? – code (1)

```
subroutine SaturatedExcessRunoff (...)
  ! Inputs: [cut for brevity]
  !
  ! Outputs:
  ! - qflx_sat_excess_surf: surface runoff due to saturated surface (mm/s)
  ! - fcov: fractional impermeable area
  ! - fsat: fractional area with water table at surface
```

What are we doing? – code (2)

```
select case (this%fsat_method)
case (FSAT_METHOD_TOPMODEL)
   call this%ComputeFsatTopmodel(..., fsat)
case (FSAT_METHOD_VIC)
   call this%ComputeFsatVic(..., fsat)
case default
   call endrun(subname//' ERROR: Unrecognized fsat_method')
end select
```

What are we doing? – code (3)

```
  do fc = 1, num_hydrologyc
     c = filter_hydrologyc(fc)
     qflx_sat_excess_surf(c) = fsat(c) * qflx_rain_plus_snomelt(c)
  end do

  do fc = 1, num_hydrologyc
     c = filter_hydrologyc(fc)
     if (col%urbpoi(c)) then
        ! send flood water flux to runoff for all urban columns
        qflx_sat_excess_surf(c) = qflx_sat_excess_surf(c) + \
             qflx_floodc(c)
     end if
  end do

end subroutine SaturatedExcessRunoff
```

What are we doing? – code (4)

```
subroutine ComputeFsatTopmodel(..., fsat)
  ! [Input argument declarations cut for brevity]
  real(r8), intent(inout) :: fsat(:) ! fractional area with water table at surface

  do fc = 1, num_hydrologyc
     c = filter_hydrologyc(fc)
     if (frost_table(c) > zwt_perched(c) .and. frost_table(c) <= zwt(c)) then
        fsat(c) = ...
     else
        fsat(c) = ...
     end if
  end do

end subroutine ComputeFsatTopmodel
```

Data structure rework for water tracers and isotopes

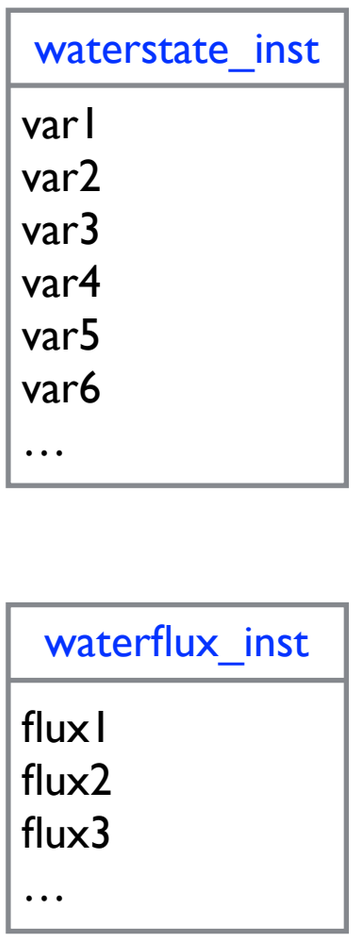
Before

waterstate_inst
var1
var2
var3
var4
var5
var6
…

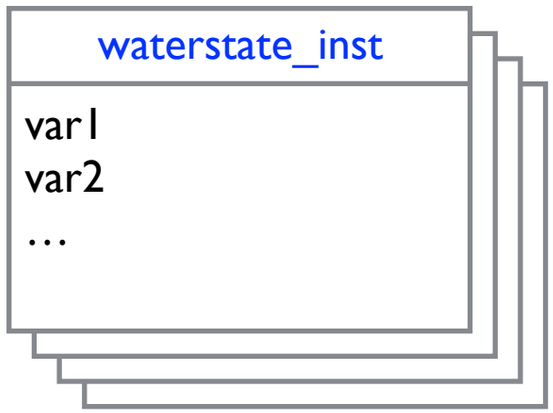
waterflux_inst
flux1
flux2
flux3
…

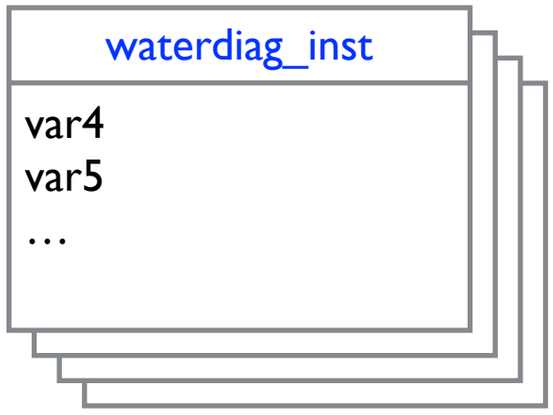
Data structure rework for water tracers and isotopes

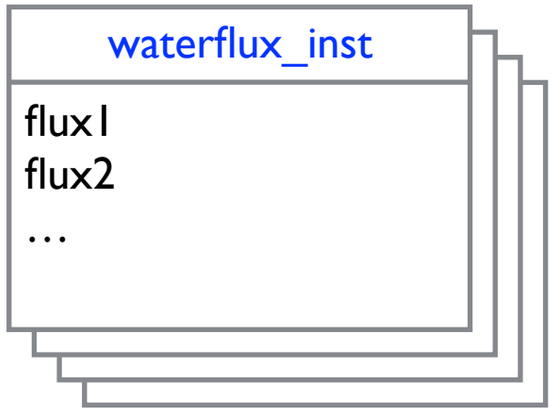
Before



After







Principles:

1. Avoid needing to change lots of science code

2. Scientists can do initial development without needing to know about tracers

Two metaphors that inspire me

Preparatory refactoring

Preparatory refactoring

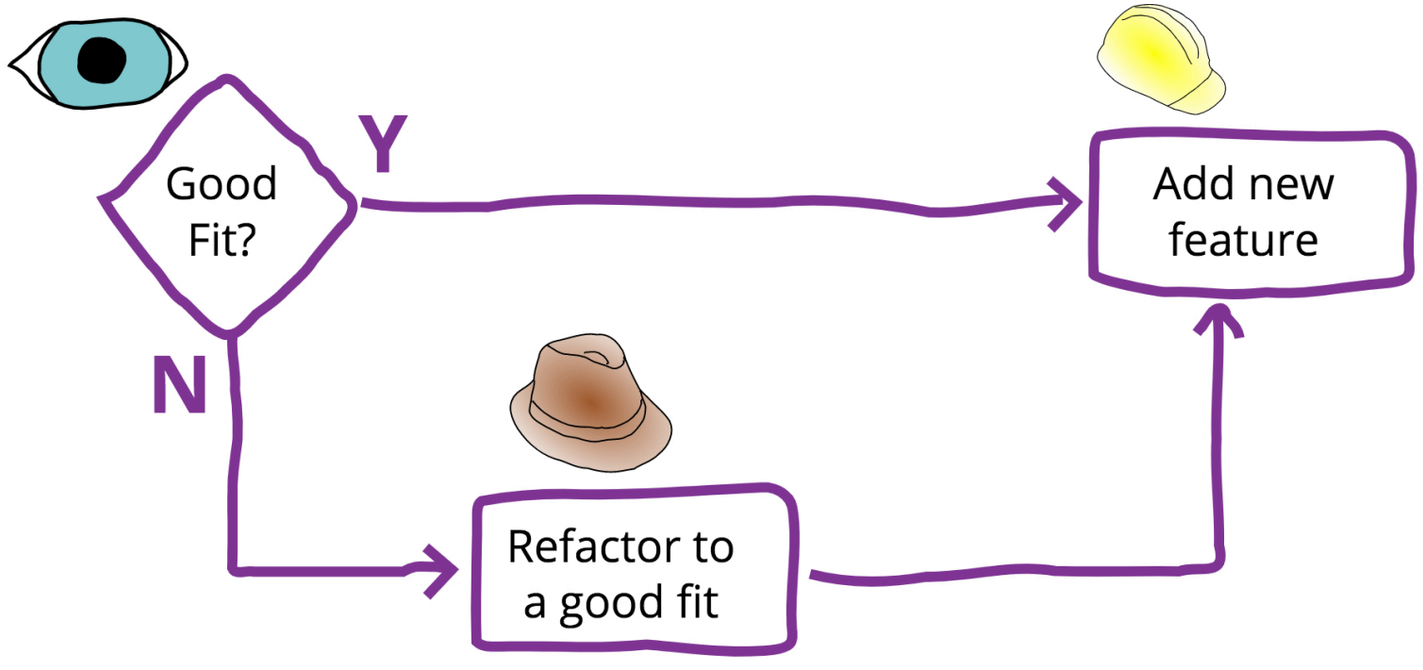
Kent Beck 
@KentBeck

Follow

for each desired change, make the change easy (warning: this may be hard), then make the easy change

4:07 PM - 25 Sep 2012

843 Retweets **1,008** Likes



https://martinfowler.com/articles/workflowsOfRefactoring

Livable code

https://brightonruby.com/2017/livable-code-sarah-mei/

You have to live here

Livable code

https://brightonruby.com/2017/livable-code-sarah-mei/

You **GET** to live here