Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

COSMO coding style and Development Workflow

Carlos Osuna, Meteoswiss
carlos.osuna@meteoswiss.ch

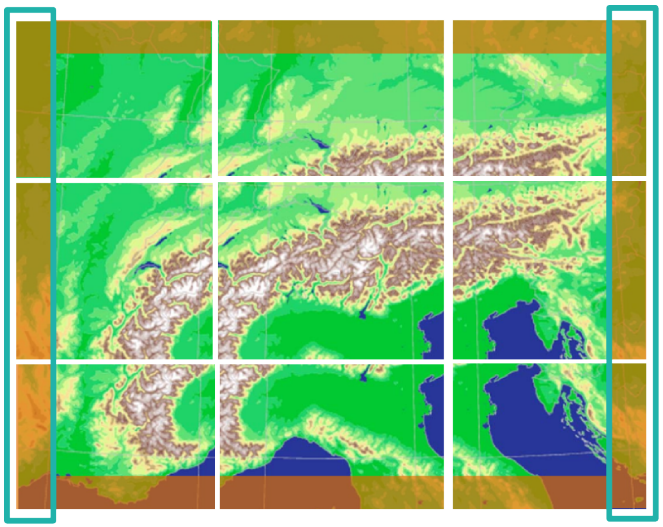
January 26, 2016

Code Modular design vs Flat Explicit Code

- COSMO coding style tends to write explicit code in flat files/modules for all functionality.
- Few new components in the last releases of COSMO change this paradigm towards a more modular design

Example from LBC

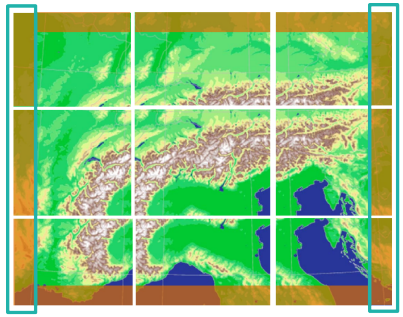
```
IF(my_cart_neigh(3) = −1) THEN
  DO k=1, ke
    DO j=1, je
      DO i=iendu, ie−1
        u (i,j,k,nnew) = z1 ∗ u_bd (i,j,k,nbd1) +
             z2 ∗ u_bd (i,j,k,nbd2)
      ENDDO
    ENDDO
  ENDDODO
ENDIF
IF(my_cart_neigh(3) = −1) THEN
  DO k=1, ke
    DO j=1, je
      DO i=1, istartu
        u (i,j,k,nnew) = z1 ∗ u_bd (i,j,k,nbd1) +
             z2 ∗ u_bd (i,j,k,nbd2)
      ENDDO
    ENDDO
  ENDDO
ENDIF
```



Example from LBC

Refactored lbc, functionality in one sub call, API self-explanatory.

```
call lbc_upoint( BCType_Interpolate, u(:,:,:, nnew), &
    BCFieldType_VectorI, 3, doEW=.True. , doNS = .False. , &
    doCorners = .True., bd1=u_bd(:,:,:,nbd1), bd2=u_bd(:,:,:,nbd2))
```



Modular Design

- Modularity $\neq$ use Fortran modules.
- Think in terms of library/functionality.

Recent examples: tracer module, mpe_io, gcl, lbc, proposal for exch_boundaries, block module.

Advantages of modular design

- Readability:

Advantages of modular design

- Readability:
  1. API/doc tells what the code in the call is doing

     call lbc_masspoint(BCType_Copy, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners, src=src(:,:,1))
     call lbc_masspoint(BCType_ZeroGradient, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners)

  2. grep lbc * (to find all BC in COSMO)
  3. Simplify by not exposing code complexity

Advantages of modular design

- Readability:
  1. API tells what the code in the call is doing

     ```
     call lbc_masspoint( BCType_Copy, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners, src=src(:,:,1) )
     call lbc_masspoint( BCType_ZeroGradient, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners )
     ```

  2. grep lbc * (to find all BC in COSMO)
  3. Simplify by not exposing code complexity
- Testability

Advantages of modular design

- Readability:
  1. API tells what the code in the call is doing

     ```
     call lbc_masspoint( BCType_Copy, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners, src=src(:,:,1) )
     call lbc_masspoint( BCType_ZeroGradient, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners )
     ```

  2. grep lbc * (to find all BC in COSMO)
  3. Simplify by not exposing code complexity
- Testability
  1. Independent functionality that can be tested: e.g. test_src_lbc.f90, metadata of traces,...

Advantages of modular design

- Readability:
  1. API tells what the code in the call is doing

     ```
     call lbc_masspoint( BCType_Copy, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners, src=src(:,:,1) )
     call lbc_masspoint( BCType_ZeroGradient, fexp(:,:,1), BCFieldType_Scalar, nlines, &
     doEW, doNS, doCorners )
     ```

  2. grep lbc * (to find all BC in COSMO)
  3. Simplify by not exposing code complexity
- Testability
  1. Independent functionality that can be tested: e.g. test_src_lbc.f90, metadata of traces,...
- Code safety
  1. less code redundancy -> less bugs
  2. explicit code spreads bugs all over the place.

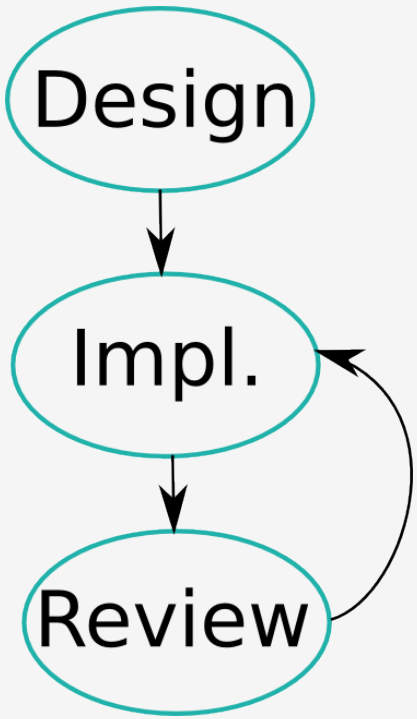
A must

- Modular/Library design makes it easier to use, but require trust:
  1. Comprehensive testing (see Pascal's talk).
  2. Expert code owner per functionality.

Design code functionalities

Number of people involved in COSMO developments is growing (from multiple institutes)...
How to coordinate/review new functionalities?

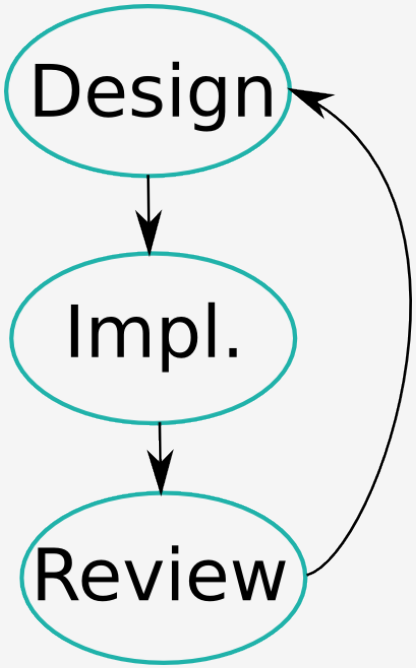
Traditional approach



Design code functionalities

How to coordinate/review new functionalities?

Sometimes



Design code functionalities

How to coordinate/review new functionalities?

Ideally