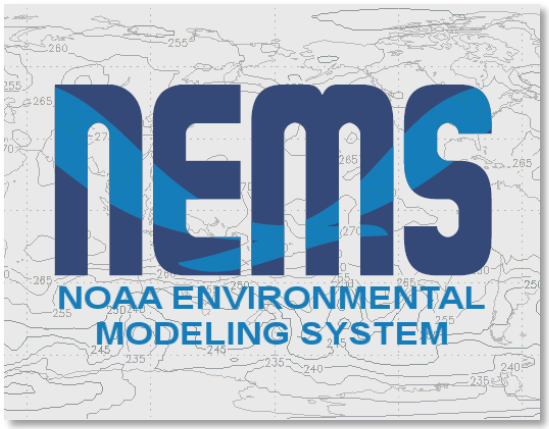
96th American Meteorological Society Annual Meeting

14th Conference on Artificial and Computational Intelligence and its Applications to the Environmental Sciences



Eugene Mirvis, IMSG @ EMC/NCEP/NWS/NOAA
Mark Iredell, EMC/NCEP/NWS/NOAA

Restructuring of the NEMS Framework: to Adapt Requirements for Independent Models Development

10–14 January, 2016 New Orleans, LA , Ernest N. Morial Convention Center

Background

NOAA Environmental Modeling System (NEMS) has been established as a framework for serving a number of operational multi-scale weather and climate prediction models based on

- *existent* NOAA Centers for Environmental Prediction (NCEP) *models*,
- a set of in-house developed *scientific and domain specific shared libraries*, tools and products, and
- taking advantage of employing the *Earth Science Modeling Framework (ESMF).*

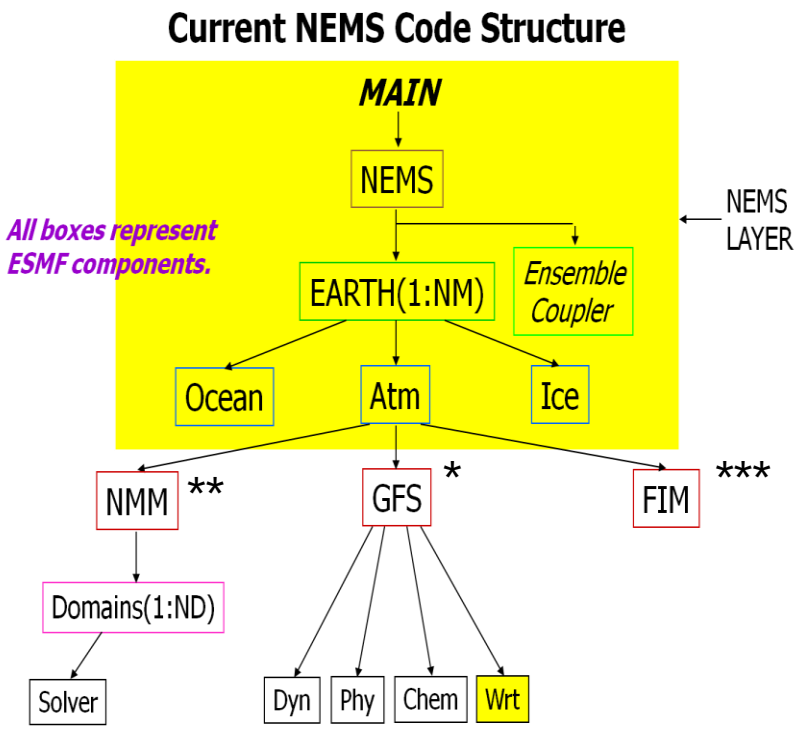
As with any framework design, the main requirements for NEMS reside in the field of the justified common code reuse and reasonable polymorphism in data type and operations.

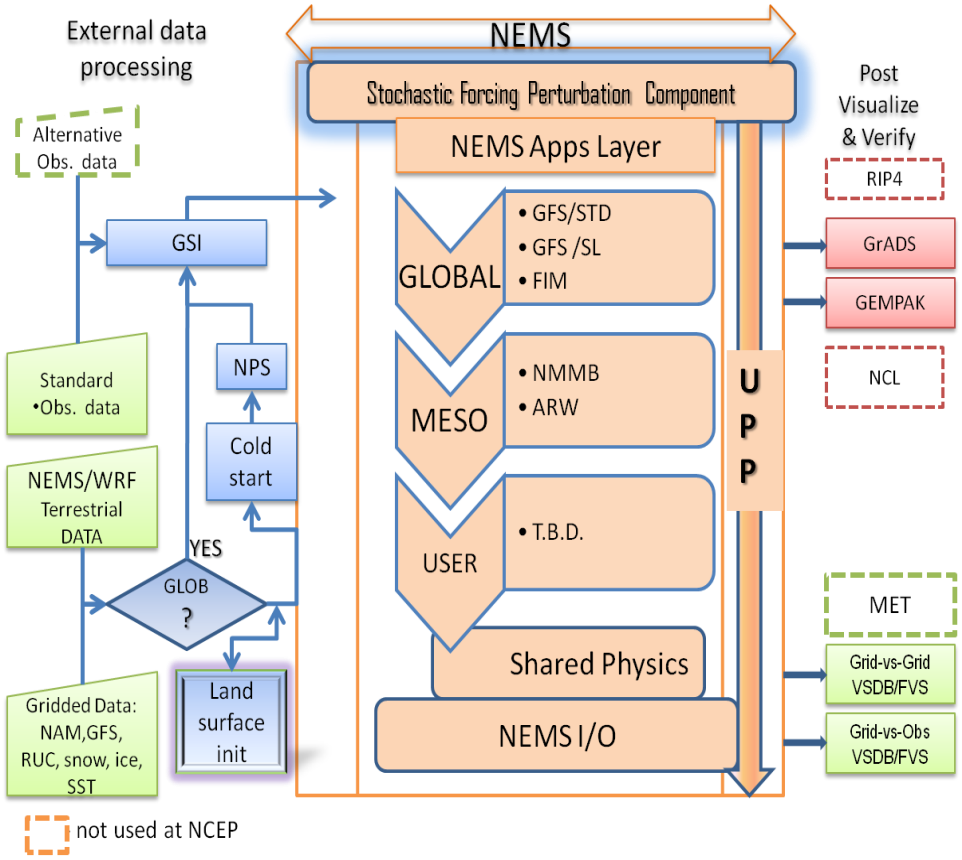
Therefore, NEMS designers initially considered a componentization of

- NEMS into the NEMS *super structure*,
- Shared *Post-processing component*,
- NEMSIO – a scalable I/O communication component;
- Available common and model specific column of *atmospheric physics collection*,
- *Internal ensembles component*,
- Integrated several available models under development
- and a number of the placeholders.

Legacy of NEMS framework design

Original NEMS Workflow Design





* - Is under development in different from NEMS / EMC repository
** - Will be separated and a subject of NGGPS dynamic core evaluation
*** - Has been always developed outside of EMC site. FIM is a Subject of ** as well

Pros, Cons and New Requirements

- Down the road, during multi-year intensive NEMS and NEMS-based models' development and integration of several *more internal and external models, new requirements have been drawn.*
- Periodic review of NEMS design pros and cons have been addressed to *question sufficiency of the initial NEMS design* & internal models code location;
- Extensive experience in NEMS development required verification of any trunk commits with a growing regression testing suite that *request running not only specific model tests but all others models as well*;
- Current NEMS *software structure code dependences lead to a necessity to rebuild "not-targeted-for-change" NEMS code components*.
- The latest active collaborative contribution to the NEMS of multi-agency *National Unified Operational Predication Capability (NUOPC)*, with capability to couple within NEMS greatest classes independently developed Earth Science models (such as Ice, Ocean etc.) *using similar mechanisms of model coupling*.

Motivations

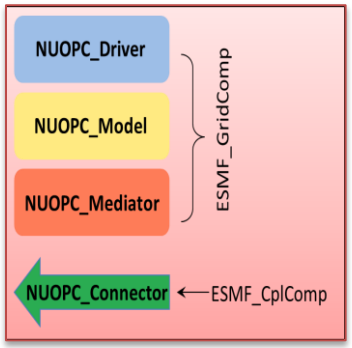
To improve NEMS and models development process:

- Unnecessary extensive different model *development teams collaboration before every commits* to the repository;
- Current needs *to checkout everything* under NEMS code;
- External links *time gap between the trunks* of the models repositories;
- Needs to *recompile models and a framework due-to "module" dependencies;*
- Constant growing number of the Regression Tests (RT) before each commit;
- *NGGPS requirements to unify physics library.*

Described efforts have been focused on NEMS restructuring solutions that can provide the *model specific teams with opportunities to conduct more independent development,* while taking full advantage of renovated NEMS's framework features reuse.

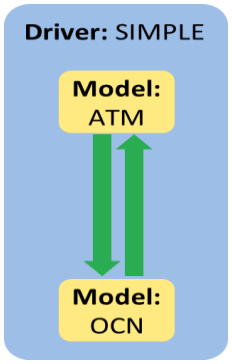
Newer NEMS Coupling standard*

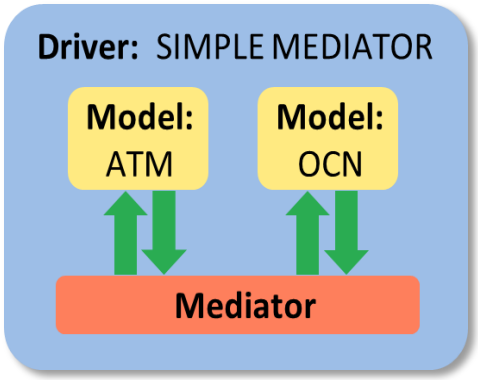
NUOPC (National Unified Operational Prediction Capability)

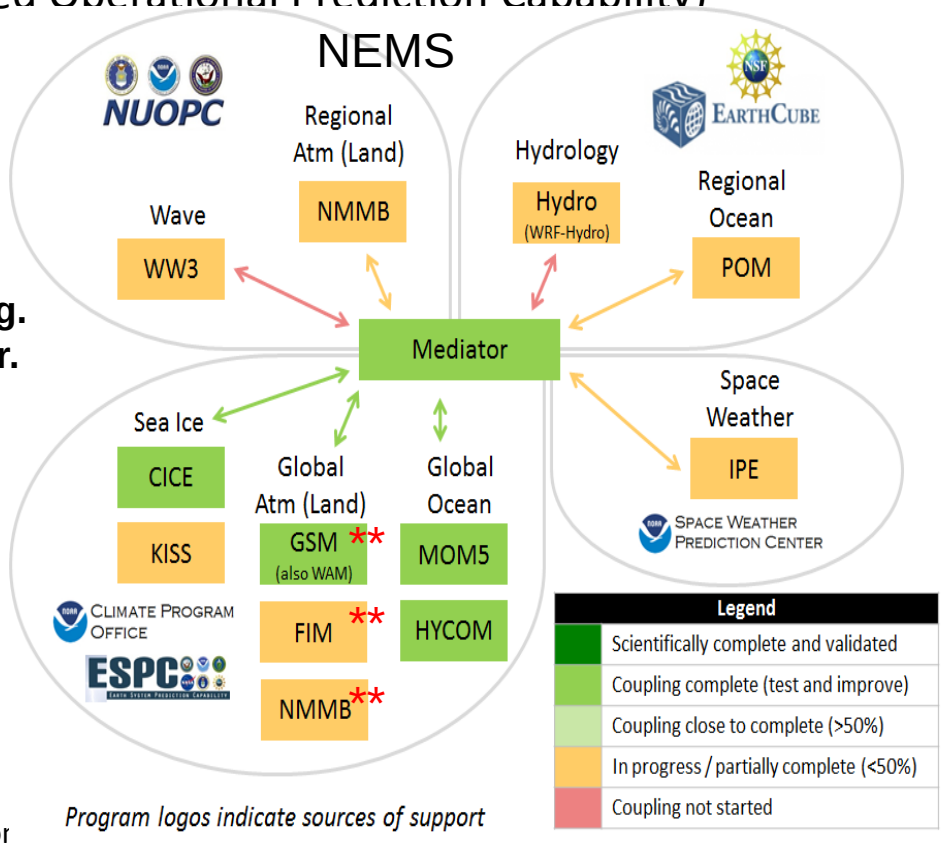


ESMF

- **Field brokering.**
- **Initialize sequence.**
- **Run sequence.**
- **Compatibility testing.**
- **Compliance checker.**







Program logos indicate sources of support

Driver - Provides a harness for Models, Mediators, and Cor the application time loop.

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

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

Connector - Connects *pairs* of components, e.g. Model to/from Model, or Model to/from Mediator, and executes simple transforms (Regrid or Redist).

** Sources of the materials on the slide is a courtesy of NUOPC group*
*** Will be revised according to dynamic core selection and NGGPS budget*

EMC specifics:

All trunk revisions must be good: NEMS Std. Regression Tests

Purpose of Regression Tests:

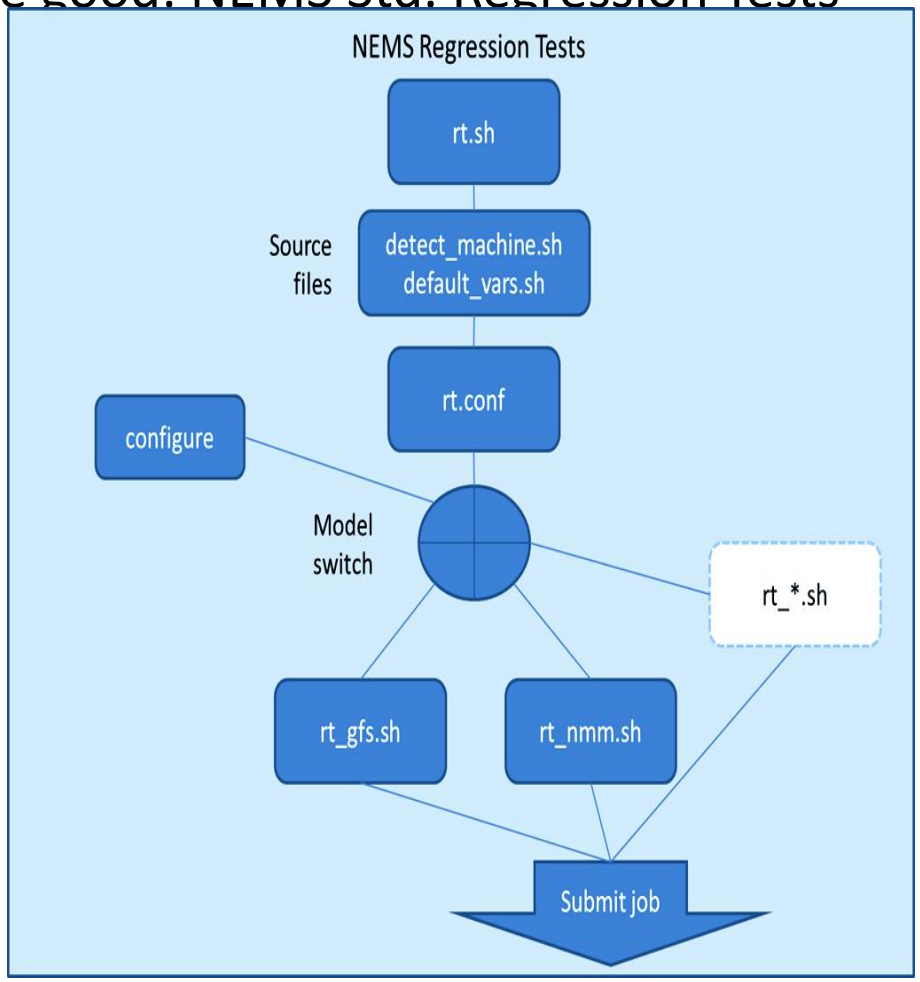
Regression testing attempts to uncover errors by partially retesting a modified program, and its intent is to provide an assurance that no additional errors were introduced into the model.

We test:

1) Bit-identical result reproducibility
2) Different domain decomposition
3) Threading
4) Different physics options
5) Timing
6) Different compilations option success and many more...

No test forecast verification (scores and skills).

The NEMS regression test suite is separated into two sets, FULL and STANDARD. It takes about 1.5 hours to finish and consists of **7 NMM-B tests**, **4 GSM tests**, 2 WAM tests, and 1 GOCART test. The FULL set is run from a cron once a week about **7 hours** to finish and has **44 tests and 2** different compilations (ESMF ver. with and without POST).



Objectives NEMS Framework restructuring

Change the file and build structures of NEMS in order:

- To be able *to isolate the common* components/parts of the models into "NEMS" library (**libnems.a)** and to build it and develop separate project, independently from the model's development, as a framework;
- To isolate *GSM/NEMS models into separate projects* and be able to develop without needs to recompile other models or **libnems.a;**
- To be able to develop each model and build its library like (**libgsm.a, libnmm.a, libmom5.a etc.**) based on the pre-installed tagged libraries of NEMS and other models, include files and the headers.
- To keep current capabilities to create **NEMS.x** for several models, invoking such as GSM & NMM, for instance.
- To separate physics library **libphys.a** from NEMS as a unified project
- Recompiling **libnems.a** should not lead to recompile a Model

Formal Typical Occurrences In The Dependencies Analysis

```
./src/atmos/gsm/makefile:include ../../conf/configure.nems
./src/atmos/gsm/makefile:          -I../../chem/gocart/${ARCH}/include/GMAO_mpeu \
./src/atmos/gsm/makefile:          -
I../../chem/gocart/${ARCH}/include/GEOSchem_GridComp
./src/atmos/gsm/makefile:MKDEPENDS = ../../../exe/mkDepends.pl

./src/atmos/gsm/phys/makefile:include ../../../conf/configure.nems
./src/atmos/gsm/phys/makefile:UTILINCS = -I../../share -I../libutil -I../../phys $(GOCARTINCS)
./src/atmos/gsm/phys/makefile:          -I../../../chem/gocart/${ARCH}/include/GMAO_mpeu
\
../../conf/make.rules
./src/atmos/post/makefile:include ../../conf/configure.nems
./src/atmos/post/makefile:UTILINCSPOST = -I../share -I../nmm/ -I../io $(POST_INC)
./src/atmos/post/makefile:UTILINCSPOSTGFS = -I../share -I../io $(POST_INC)
./src/atmos/post/makefile:UTILINCSPOSTNMM = -I../share -I../nmm $(POST_INC)
./src/atmos/post/makefile:UTILINCSPOSTSTUB = -I../share
```

Moving to the pre- build libraries based NEMS structure

Current File Structure

./.svn
./ENS_Cpl
./atmos
./conf
./chem
./ESMFVersionDefine_ESMF_
./ESMFVersionDefine_ESMF_
./module_EARTH_GENERIC_(
./module_NEMS_GRID_COM
./module_EARTH_GRID_COM
./ESMFVersionDefine_ESMF_
./ESMFVersionDefine_ESMF_
./esmf_version
./module_NEMS_INTERNAL_
./module_EARTH_INTERNAL_
./MAIN_NEMS.F90
./ESMFVersionLogic.h
./module_MEDIATOR.F90
./makefile
./ESMFVersionDefine_ESMF_

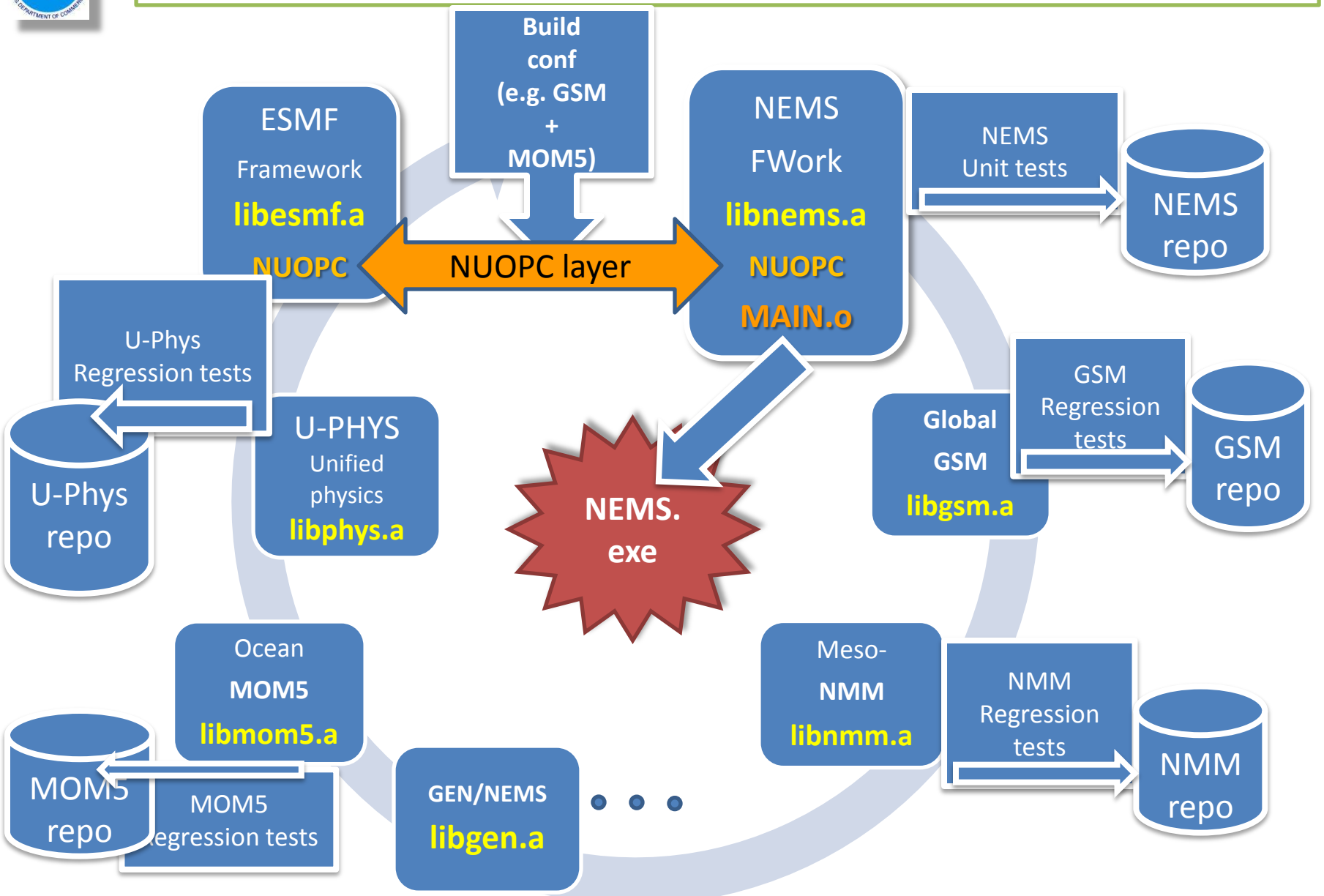
```
`--atmos
|  `--fim
|  `--gen
|  `--gsm
|  |  `--dyn
|  |  `--libutil
|  |  `--phys
|  `--io
|  `--nmm
|  `--phys
|  `--post
|  `--share
`--chem
```

Prebuild lib structure outcome

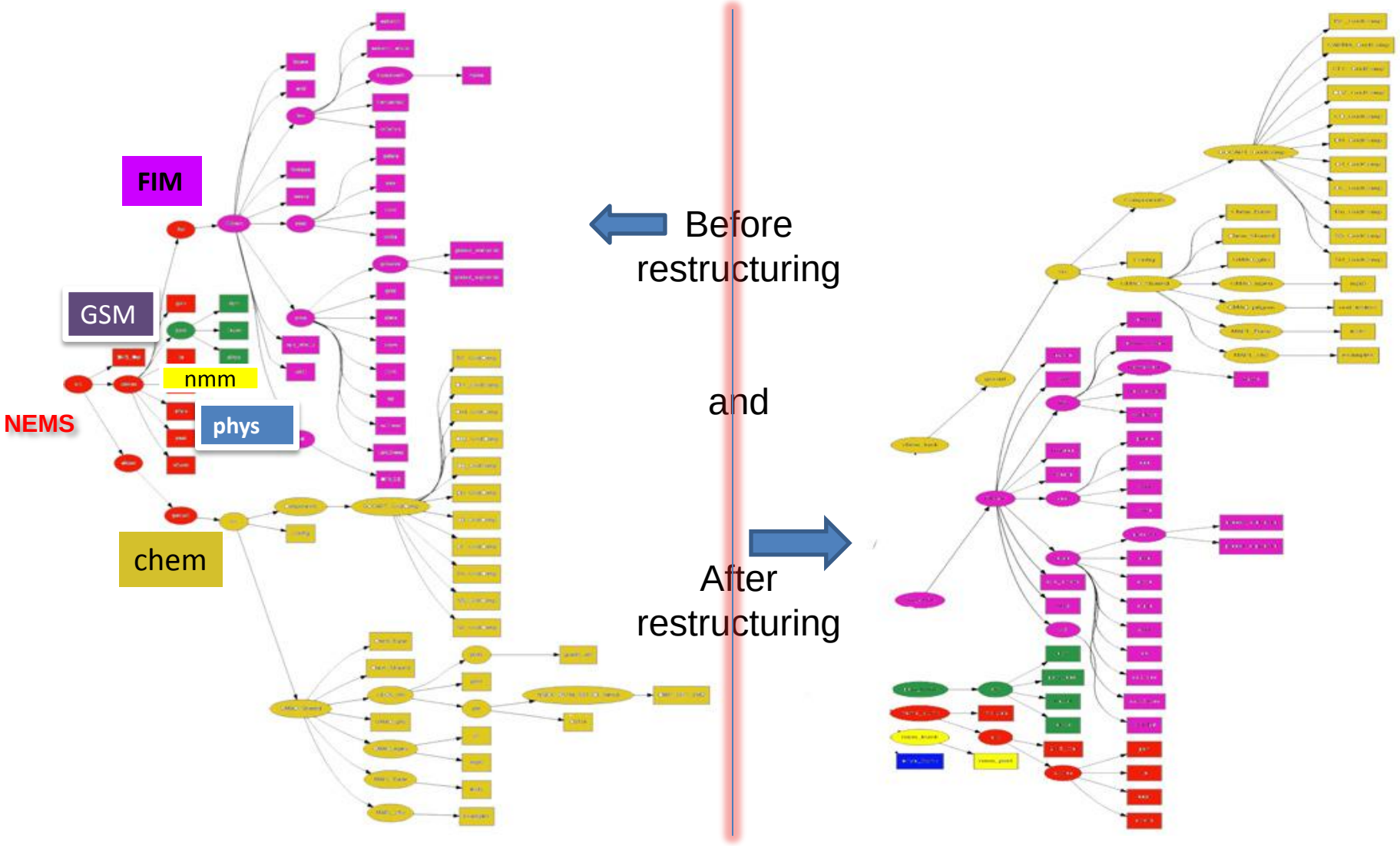
./include
./include/ESMFVersionDefine_ESMF_3.h
./include/ES
./include/ES
./include/ES
./include/ES
./include/ES
./include/M
./include/m
./include/m
./include/m
./include/N
./include/N
./include/A
./mod
./mod/shar
./mod/post
./mod/ENS_
./mod/mod
./mod/mod
./mod/mod
./mod/phys
./mod/io
./mod/chem
./mod/NEMS.mk
./lib
./lib/libnems.a
./lib/libphys.a
./lib/libGOCART_GridComp.a
./lib/libgfs.a

```
`--include
`--lib
`--mod
|  `--chem
|  |  `--Chem_Base
|  |  `--GEOSchem_GridComp
|  |  `--GMAO_mpeu
|  `--ENS_Cpl
|  `--io
|  `--phys
|  `--post
|  `--share
```

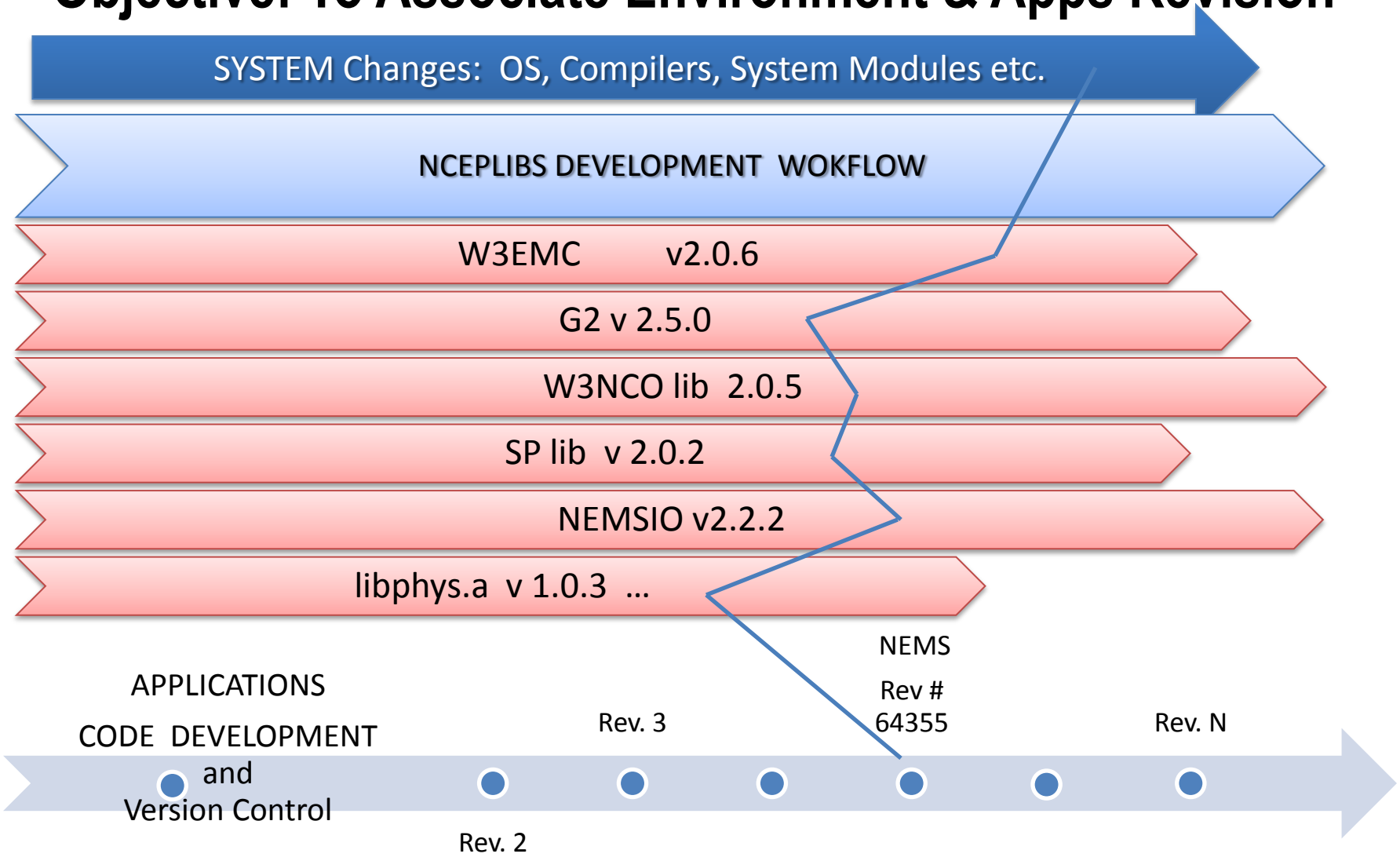
Aimed : NEMS Executable Build After Restructuring



Dependencies Analysis, Changes & Restructured of NEMS directories/repositories



Objective: To Associate Environment & Apps Revision



NCEPLIBS builds & naming conventions.

User Libraries Build Structure Standard

$NCEPLIBS/

lib/

<lib_name>/

<lib_version>/

<compiler_OPT>/

<compiler_debug /

lib<lib_name><ver><precision>.a

include/…

data01/…

Example:

/contrib/nceplibs/

lib/

w3emc/

v2.0/

intel_O3/

libw3_4.a

intel_gO0/

libw3emc_2.2.0_4.a

LIBS Environment Parameters naming Standard

<LIB_NAME>_<TYPE of files><PRECISION><ADD_ATTRIBUTES>
for instance: ‘‘SP_LIBd’‘ or ‘‘SIGIO_INC4’‘

<LIB_NAME> :: is capitalized library name (e.g. BUFR, SP, IP, NEMSIO etc.)

<LIB_NAME>_VER :: the version of the library, consistent with EMC NCEPLIBS SVN, starts with ‘‘v’‘- small case and to follow triadic: <release>.<revision>.<patch>, separated by ‘‘.’‘: v10.2.5 or v2.0.1 etc.

<TYPE of files> ::

1. ‘‘SRC’‘ – location of the source codes
2. ‘‘INC’‘ – location of the ‘‘include’‘ files
3. ‘‘LIB’‘ – location of the libraries (archives - *.a files)

<PRECISION> :: the latter or number, referring to the precision used for compilation of the default types, such as:
‘‘4’‘ - compilation with < -r4 -i4 >
‘‘8’‘ – compilation with <-r8 –i8>
‘‘d’‘- compilation with < -i4 –r8>, ‘‘s’‘ etc.

<ADD_ATTRIBUTES> ::
Any additional attributes of the library, such as an exception from naming convention or future naming convention

FEE: From Sys->3rd Libs> Your - Libs -> NEMS

Availability of the user & 3rd party libs and their Modulfiles

```
_> module load bacio crtm
_> env
BACIO_LIB4=/nwprod/lib/bacio/v2.0.1/libbacio_v2.0.1_4.a
BACIO_LIB8=/nwprod/lib/bacio/v2.0.1/libbacio_v2.0.1_8.a
BACIO_SRC=/nwprod/lib/bacio/v2.0.1/sorc
BACIO_VER=v2.0.1
CRTM_INC=/nwprod/lib/crtm/v2.1.3/incmod/crtm_v2.1.3
CRTM_LIB=/nwprod/lib/crtm/v2.1.3/libcrtm_v2.1.3.a
CRTM_SRC=/nwprod/lib/crtm/v2.1.3/src
CRTM_VER=v2.1.3
```

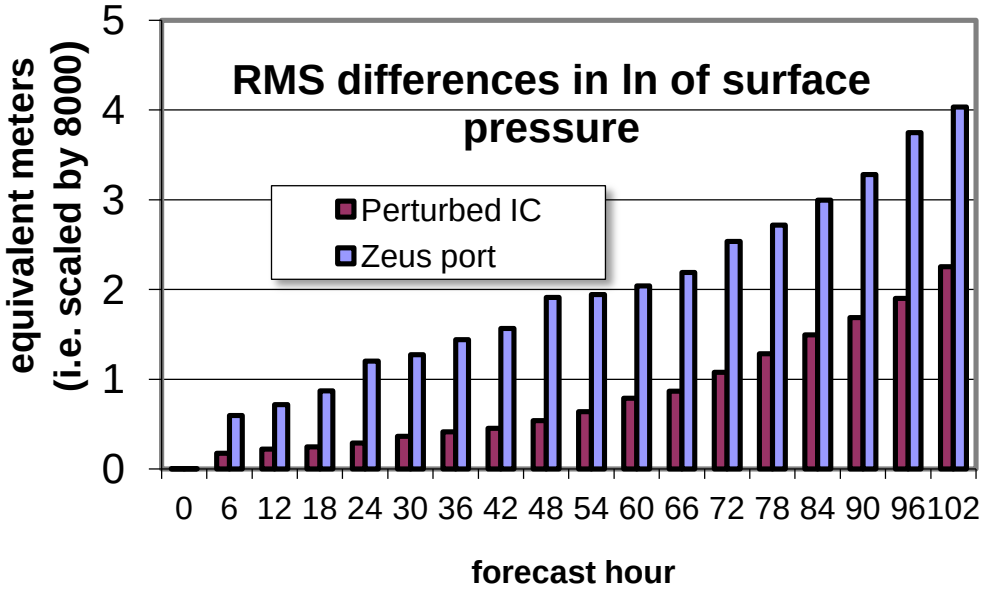
Apps Dependent Part of the Apps Makefile

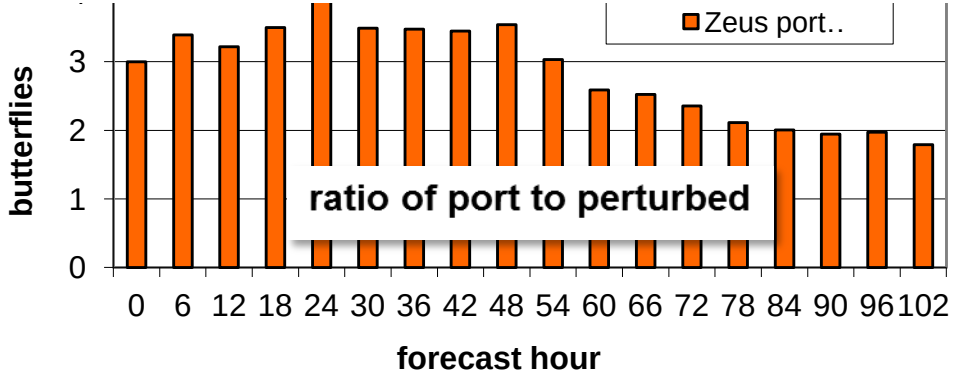
```
EXTLIBS =      $(NEMSIO_LIB) $(BACIO_LIB4) $(W3NCO_LIBd)  \
$(W3EMC_LIBd) $(SP_LIBd) $(NETCDF) $(ESMF)

EXTLIBS_POST = $(NEMSIO_LIB) $(POST_LIB) $(W3NCO_LIBd) \
      $(W3EMC_LIBd) $(XMLPARSE_LIB) $(G2_LIB4) $(G2TMPL_LIB) \
      $(JASPER_LIB) $(PNG_LIB) $(Z_LIB) $(XMLPARSE_LIB) \
      $(SIGIO_4) $(SFCIO_4) $(CRTM_LIB) \
      $(ESMF_LIB) $(NETCDF_LIB) $(SYS_LIB)
```

NEMS_v1.3.7 Modulefile
(Auto- generated)

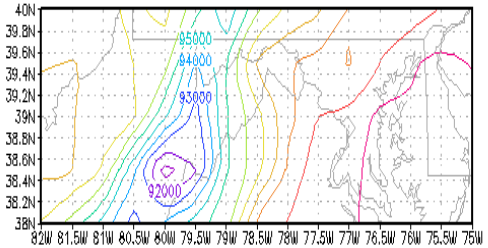
```
#%Module
##        autogenerated via 'GetAppsEnv' ##
     module purge
     module load intel/15.1.133
     module load impi/5.0.3.048
     module load esmf/6.3.0rp1
     module load netcdf/4.3.0
     module load nemsio/v2.2.1
     module load bacio/v2.0.1
     module load w3nco/v2.0.6
     module load w3emc/v2.0.5
     module load sp/v2.0.2
     module load xmlparse/v2.0.0
     module load g2/v2.5.0
     module load g2tmpl/v1.3.0
     module load jasper/v1.900.1
     module load png/v1.2.44
     module load z/v1.2.6
     module load sigio/v2.0.1
     module load sfcio/v1.0.0
     module load crtm/v2.1.3
```

Butterfly Tests of "Functional Equivalence"

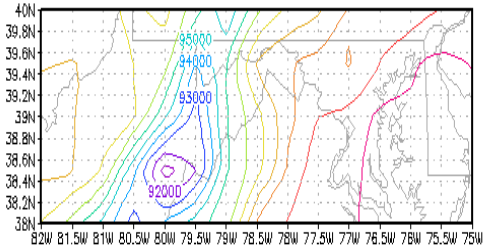




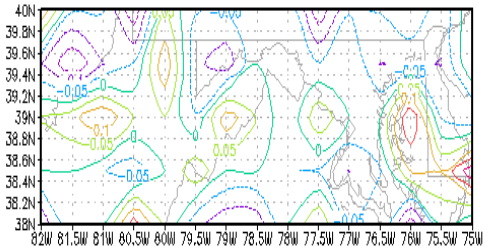
In this example, the perturbation is applied only to the initial surface pressure for the 2011-08-31-12z GFS



Original surface pressure (Pa)

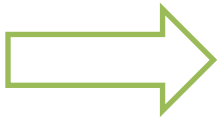


Perturbed surface pressure (Pa)



Difference in surface pressure (Pa)

Thank you!

Additional Slides 

Additional NEMS Restructuring Points

1. Anything (code, data, build scripts and makefiles) in "nems" should be independently develop (changed) from anything using nems, buid system, ESMF, ncep and 3rd party libs
2. The result of the build (configuration) should be stored in the **NEMS installs** and used in components, based on it.
3. All models would be using only build NEMS, pre- build physics , chem
4. Common NEMS components can not be replicated in the "models" directories, however anything specific in configure have to have model prefix and NEMS should not be forced to adapt it, unless decided for other models.
5. Before building model, model developer will load NEMS.versioned environment module and run installation script "Install" which will use common var $NEMSLIBS
6. Exe will be build on the model side. Also, exe can be build on the code running site, with all needed versioned libs pre-build
7. Repository projects ,would be addressed separately with no external links
8. After nems build all compiled in 1 libnems.a, except ../mod directory in vertical structure, needed to build Models
9. Build of the executable would depend from atm.configure
10. Each Nems build would have only 1 Esmf and Libs configuration. We need to decide how many to build and support.
11. In spite of the specifics all NEMS I/O – should be still a part of "NEMS"
12. ENS_Cpl is a part of NEMS
13. Post - based on the use (where)
14. Some common "share" is still allowed, like "machine" "constants" and some not resolved yet

subroutine to perturb any field (mersenne_twister is in w3lib)

```
subroutine pert6(n,f)
  use mersenne_twister
  implicit none
  integer,intent(in):: n
  real,intent(inout):: f(n)
  integer i
  real r
  do i=1,n
    r=random_gauss_f()
    f(i)=f(i)+r*abs(f(i)/1.e6)
  enddo
end subroutine
```

EMC Functional Equivalence Criteria

To develop a formal functional equivalence test so that a minor restructuring or porting of the code causing non-bit-identical butterfly differences can pass the regression tests.

- **Select key fields from the import state F_i**
- **Run control run C of application on control environment**

- Run perturbed run P from perturbed import state on control environment.
- Select key fields from the export states C_j (control run) and P_j (perturbed run).
  a. may include a selection of output times
- Determine area weight w(x)
- Compute the butterfly vector $\beta_j = \left[\frac{w_j P_j^2 - C_j^2}{\overline{w}} \right]^{\frac{1}{2}}$
- Butterfly vector will have units of original export fields
- Now run control run but on different environment D
- Compute difference vector δ_j like butterfly vector but using D instead of P.
- Compute dimensionless difference butterfly number vector $\nu_j = \delta_j / \beta_j$
- For a bit-identical requirement, ν_j must be zero
- For a functionally equivalent requirement, ν_j must be largely less than 10

module_ATM_GRID_COMP.F90

```
     51       USE module_NMM_GRID_COMP,ONLY: NMM_REGISTER
     52       USE module_GFS_GRID_COMP,ONLY: GFS_REGISTER
     53       USE module_FIM_GRID_COMP,ONLY: FIM_REGISTER
     54       USE module_GEN_GRID_COMP,ONLY: GEN_REGISTER   ! For the "Generic Core" gridded
component.
…
…
  1087         CASE('nmm')
   1088           CALL ESMF_GridCompSetServices (atm_int_state%CORE_GRID_COMP   &
   1089                                         ,NMM_REGISTER                   &
   1090                                         ,rc=RC)
   1091 !
   1092         CASE('gfs')
   1093           CALL ESMF_GridCompSetServices (atm_int_state%CORE_GRID_COMP   &
   1094                                         ,GFS_REGISTER                   &
   1095                                         ,rc=RC)
   1096 !
   1097         CASE('gsm')
   1098           CALL ESMF_GridCompSetServices (atm_int_state%CORE_GRID_COMP   &
   1099                                         ,GFS_REGISTER                   &
   1100                                         ,rc=RC)
```

Deployment Dependencies Reduction for GSM

NEMS: module_ATM_GRID_COMP.F90

…

Call CalledGSM_Register(GFS_GRID_COMP,RC_REG)

…

```
CASE('gfs')
      CALL ESMF_GridCompSetServices (atm_int_state%CORE_GRID_COMP
&
                        ,GFS_REGISTER          &
                        ,rc=RC)
```

…

GSM: module_GFS_GRID_COMP.F90:

…

SUBROUTINE GFS_REGISTER(GFS_GRID_COMP,RC_REG)

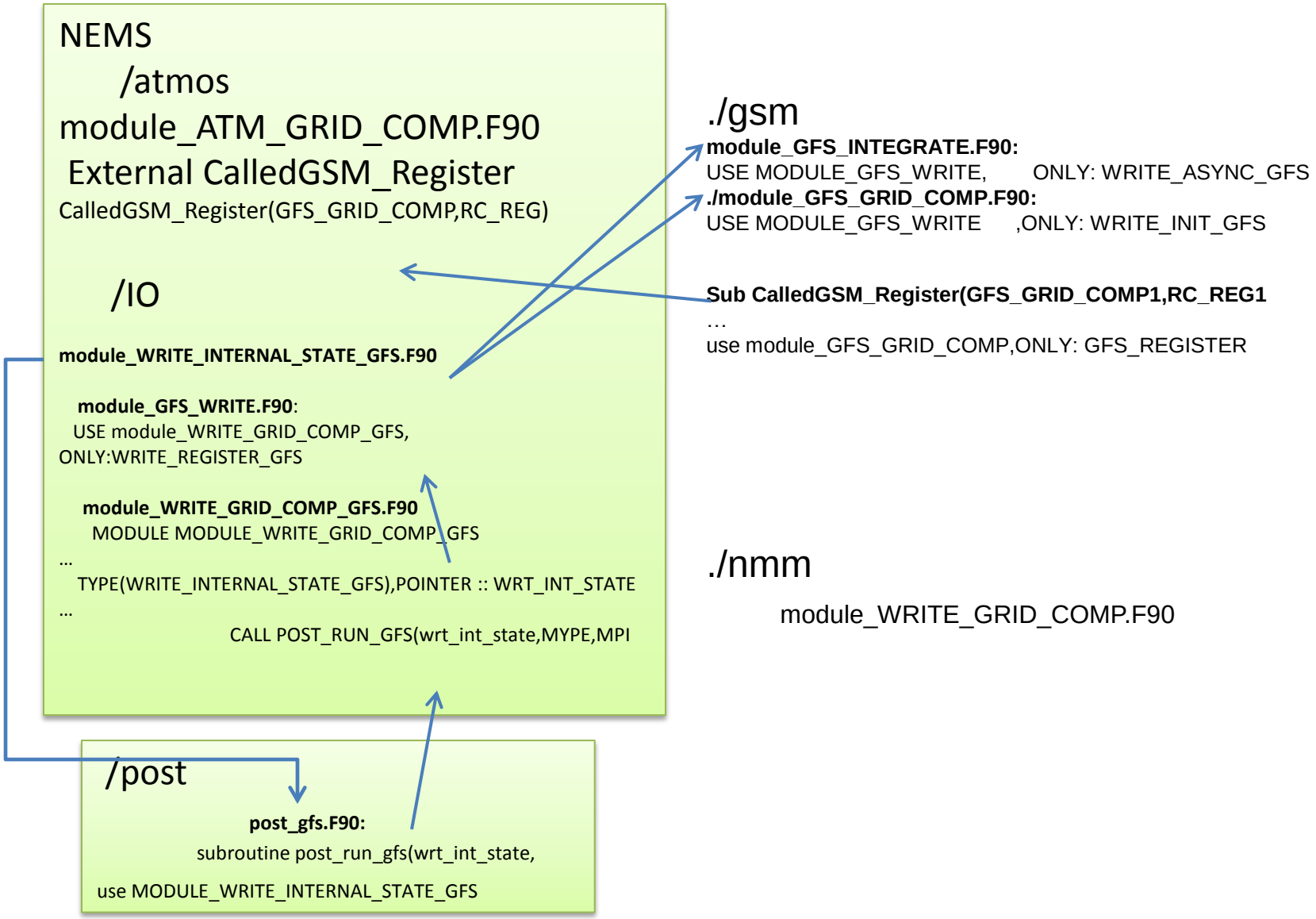
```
!-------------------------------------------------------------------------
!***  Register the GFS gridded component's initialize, run, and finalize
!***  routines.
!-------------------------------------------------------------------------
!
   TYPE(ESMF_GridComp), INTENT(INOUT) :: GFS_GRID_COMP  !<-- GFS gridded component
!
   INTEGER,INTENT(OUT) :: RC_REG                        !<-- Return code for register
!
!-------------------------------------------------------------------------
!***  LOCAL VARIABLES
!-------------------------------------------------------------------------
!
   INTEGER :: RC
```

```
17.4.18 ESMF_GridCompSetServices - Call user routine to register
GridComp methods
INTERFACE:
  recursive subroutine ESMF_GridCompSetServices(gridcomp, &
    userRoutine, userRc, rc)
ARGUMENTS:
    type(ESMF_GridComp), intent(inout)       :: gridcomp
    interface
     subroutine userRoutine(gridcomp, rc)
      use ESMF_CompMod
      implicit none
      type(ESMF_GridComp)      :: gridcomp ! must not be optional
      integer, intent(out)     :: rc       ! must not be optional
     end subroutine
    end interface
 -- The following arguments require argument keyword syntax (e.g. rc=rc). --
    integer,          intent(out), optional :: userRc
    integer,          intent(out), optional :: rc
```

GSM: subroutine CalledGSM_Register(GFS_GRID_COMP1,RC_REG1)

```
      use esmf_mod
      use ESMF_CompMod
      use module_GFS_GRID_COMP,ONLY: GFS_REGISTER
!***  Register the GFS gridded component's initialize, run, and finalize
!***  routines.
TYPE(ESMF_GridComp)  :: GFS_GRID_COMP1  !<-- GFS gridded
INTEGER,INTENT(OUT) :: RC_REG1    !<-- Return code for register
! ---------------------------------------------------------------------
      call GFS_REGISTER(GFS_GRID_COMP,RC_REG)
                  GFS_GRID_COMP1=> GFS_GRID_COMP
                  RC_REG1=> RC_REG
return
end subrutine CalledGSM_Register
```

NEMS<->GSM /io and /post



NEMS<->NMM /io and /post

NEMS

/atmos

module_ATM_GRID_COMP.F90

External CalledGSM_Register

CalledGSM_Register(GFS_GRID_COMP,RC_REG)

/IO

./nmm

module_DOMAIN_GRID_COMP.F90

module_WRITE_ROUTINES

module_WRITE_GRID_COMP.F90

CALL POST_RUN_NMM(wrt_int_state, MYPE,MPI_COMM_COMP,

module_WRITE_INTERNAL_STATE.F90

TYPE(WRITE_INTERNAL_STATE),POINTER :: WRT_INT_STATE

/post

post_nmm.F90:

subroutine post_run_nmm(wrt_int_state,

use MODULE_WRITE_INTERNAL_STATE