# **natESM Sprint MESSy/ComIn**
# Enabling Messy to use ComIn and ICON/ComIn

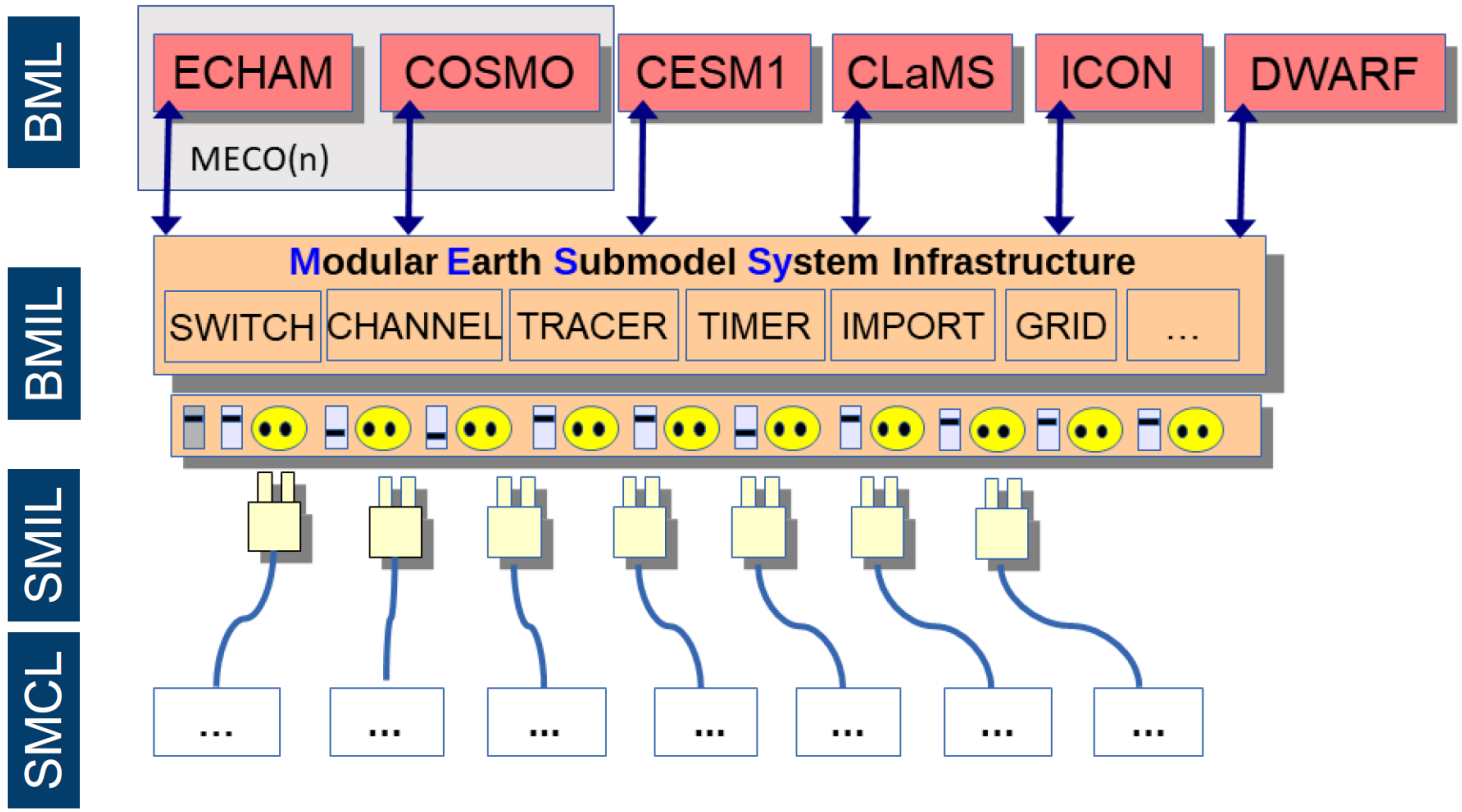
**Kerstin Hartung, Bastian Kern, Patrick Jöckel, Astrid Kerkweg (DLR-PA, FZJ IEK-8)**

**And Wilton Loch (DKRZ)**

**Supported by natESM**

# MESSy and its basemodels

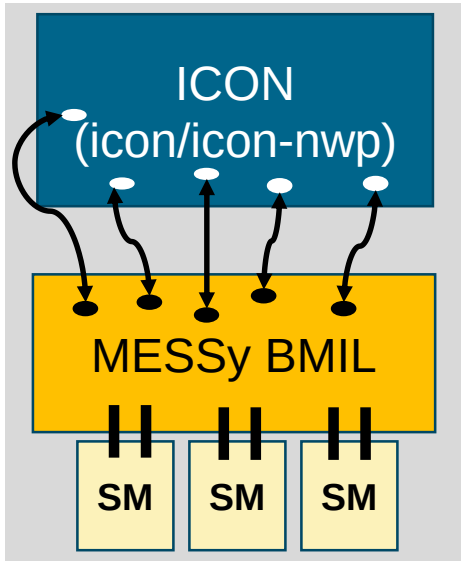
## MESSy code structure

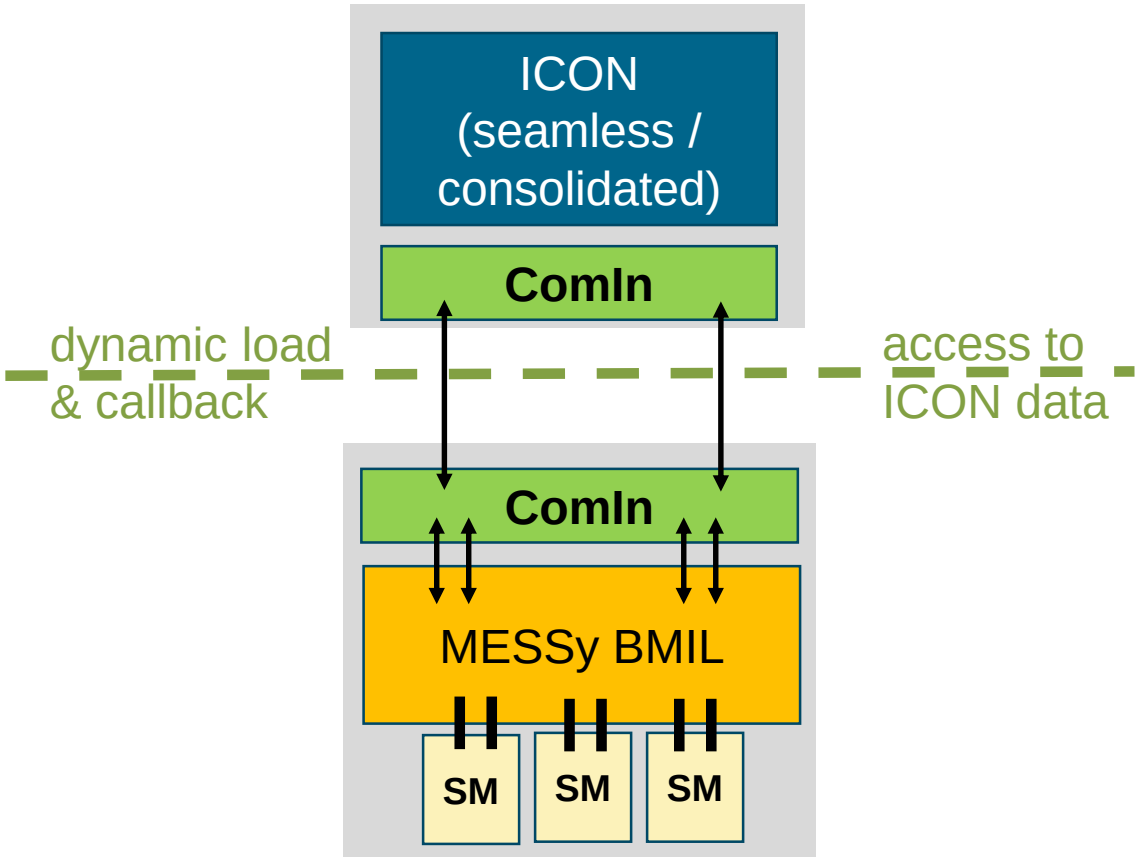


# ICON/MESSy and ComIn

**Goal:** reduced effort to keep up-to-date with ICON developments

**Current status:**



# General plugin: where to start?

```
SUBROUTINE plugin_setup()

  version = comin_setup_version_info

  CALL comin_request_add_var(t_comin_var_descriptor(jg=1, name="myvariable"),
       t_comin_var_metadata(…))

  CALL comin_callback_register(EP_SECONDARY_CONSTRUCTOR, my_constructor, ierr)

  p_global => comin_descrdata_get_global()

END SUBROUTINE plugin_setup
```

request add. ICON variable

register callback

get descr. data

```
SUBROUTINE my_constructor()

  CALL comin_var_get(EP_BEFORE_WRITE_OUTPUT, var_descr, FLAG_READ, press)

END SUBROUTINE my_constructor
```

get variable pointers

# Implementation notes (not complete)

**If already using ICON within a plugin code, some interaction might become more complex…**

- some components of p_patch not available via ComIn
- not possible to call plugin at block-loop level
- variables shared at current time-level
- MPI functionality cannot be used from ICON

**… but with ComIn the interaction with ICON also simplifies**

- flexible addition of data fields and tracers
- connection to further software easily possible, e.g. YAC, and data fields from other plugins can also easily be accessed
- some auxiliary routines (and descriptive data) named more intuitively

# natESM MESSy/ComIn sprint

## Goal:

- get to a version which runs as soon as possible
- original and new implementation need to co-exist during development and especially for evaluation

## Approach:

- iterative implemention updates checking functionality of intermediate setups
- workarounds/patches if either MESSy or ComIn do not support functionality at the moment
- document steps and challenges for other plugins

# natESM MESSy/ComIn sprint

## 1. Shared library of MESSy and ComIn

- include ICON on MESSy side as well, e.g. to be able to use MPI routines
- reading of some ICON namelist to generate grid for this helper ICON

## 2. Control module with primary constructor

- get information on descriptive data
- gather MESSy tracer metadata and request tracers from ComIn (with lconv, lturb, …)

# natESM MESSy/ComIn sprint: implementation

```
SUBROUTINE messy_comin_setup()

   CALL init_icon_mpi_grid()

   CALL comin_callback_register(EP_SECONDARY_CONSTRUCTOR,
          messy_comin_constructor, ierr)

   CALL messy_setup()
   CALL messy_initialize

   CALL messy_new_tracer

   CALL messy_request_tracers

END SUBROUTINE messy_comin_setup
```

**some workarounds**

**register callback**

**initialize MESSy and MESSy submodels**

**gather info on new tracers, receive descr. data**

**call request_add_var**

# natESM MESSy/ComIn sprint: implementation

```
SUBROUTINE messy_comin_setup()

   CALL init_icon_mpi_grid()

   CALL comin_callback_register(EP_SECONDARY_CONSTRUCTOR,
          messy_comin_constructor, ierr)

   CALL messy_setup()
   CALL messy_initialize

   CALL messy_new_tracer

   CALL messy_request_tracers

END SUBROUTINE messy_comin_setup
```

**some workarounds**

**register callback**

**initialize MESSy and MESSy submodels**

**gather info on new tracers, receive descr. data**

**call request_add_var**

# natESM MESSy/ComIn sprint

## 1. Shared library of MESSy and ComIn

- include ICON on MESSy side as well, e.g. to be able to use MPI routines
- reading of some ICON namelist to generate grid for this helper ICON

## 2. Control module with primary constructor

- get information on descriptive data
- gather MESSy tracer metadata and request tracers from ComIn (with lconv, lturb, …)

## 3. Call secondary constructor as well (work-in-progress)

- receive all tracer field metadata
- loop through ComIn variable list and get pointers to fields
- expand MESSy metadata (e.g. by representation (vertical and horizontal grid layout))
- initialize MESSy memory

# natESM MESSy/ComIn sprint: implementation

```
SUBROUTINE messy_comin_setup()

   CALL init_icon_mpi_grid()

   CALL comin_callback_register(
    EP_SECONDARY_CONSTRUCTOR,
    messy_comin_constructor, ierr)

   CALL messy_setup()
   CALL messy_initialize

   CALL messy_new_tracer

   CALL messy_request_tracers

END SUBROUTINE messy_comin_setup
```

```
SUBROUTINE messy_comin_constructor()

  CALL messy_tracer_metadata_comin
```

**receive tracer metadata**

```
  CALL messy_init_memory(1)
```

**receive tracer fields**

```
  CALL messy_init_memory(3)
```

**get pointers to ICON fields, init SM memory**

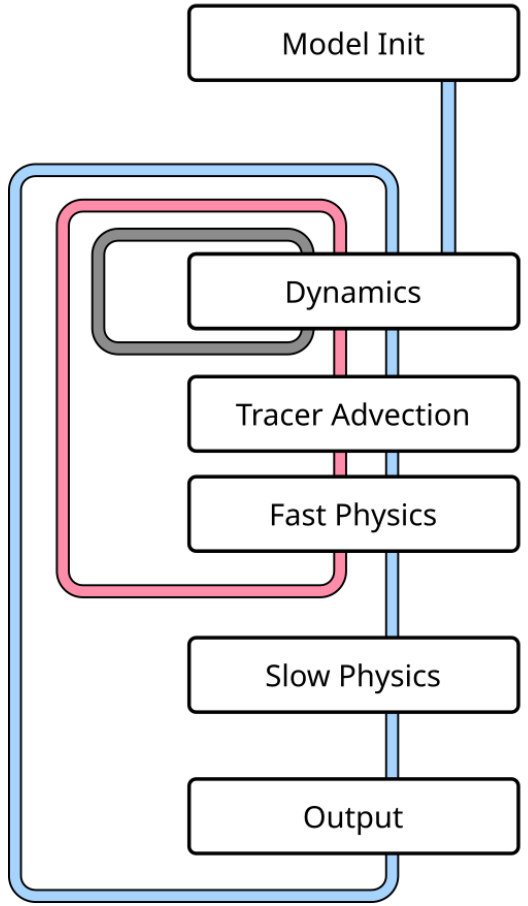
```
END SUBROUTINE messy_comin_constructor
```

# natESM MESSy/ComIn sprint: next steps

## 4. Time loop (upcoming)

- add calls to current entry points in MESSy
- calculate process tendencies (before and after entry points)
- workaround for access to currently routine-local fields (e.g. below currently supported block-loop level)

Final test and evaluation of implementation




Prill et al., ICON tutorial

# Impressum

| | |
|---|---|
| Thema: | **natESM sprint MESSy/ComIn** |
| Datum: | 15.11.2023 |
| Autor: | Kerstin Hartung |
| Institut: | DLR-PA-ESM |
| Bildcredits: | Alle Bilder „DLR (CC BY-NC-ND 3.0)“ |