



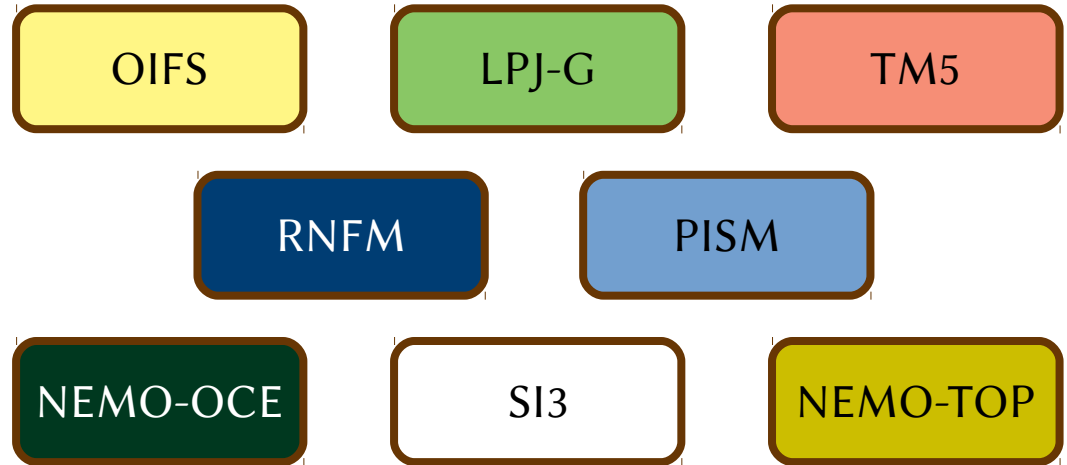
Flexible coupling of ESM components for practical use

CW2020, Uwe Fladrich, SMHI

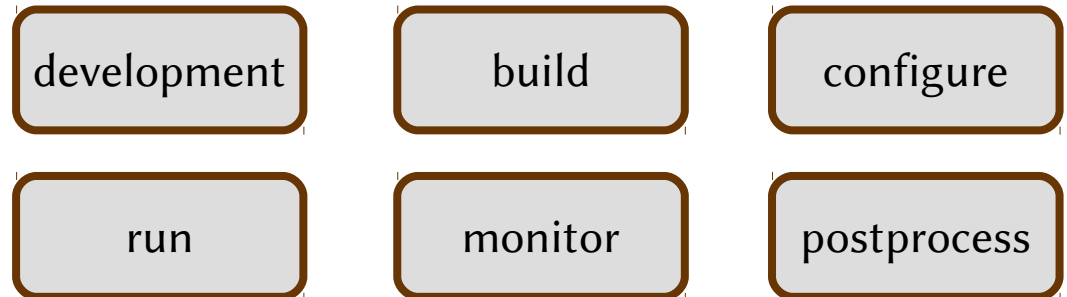
Acknowledging contributions by the EC-Earth community, particularly AWI and BSC.

ECE4 architecture and work flow

Components:



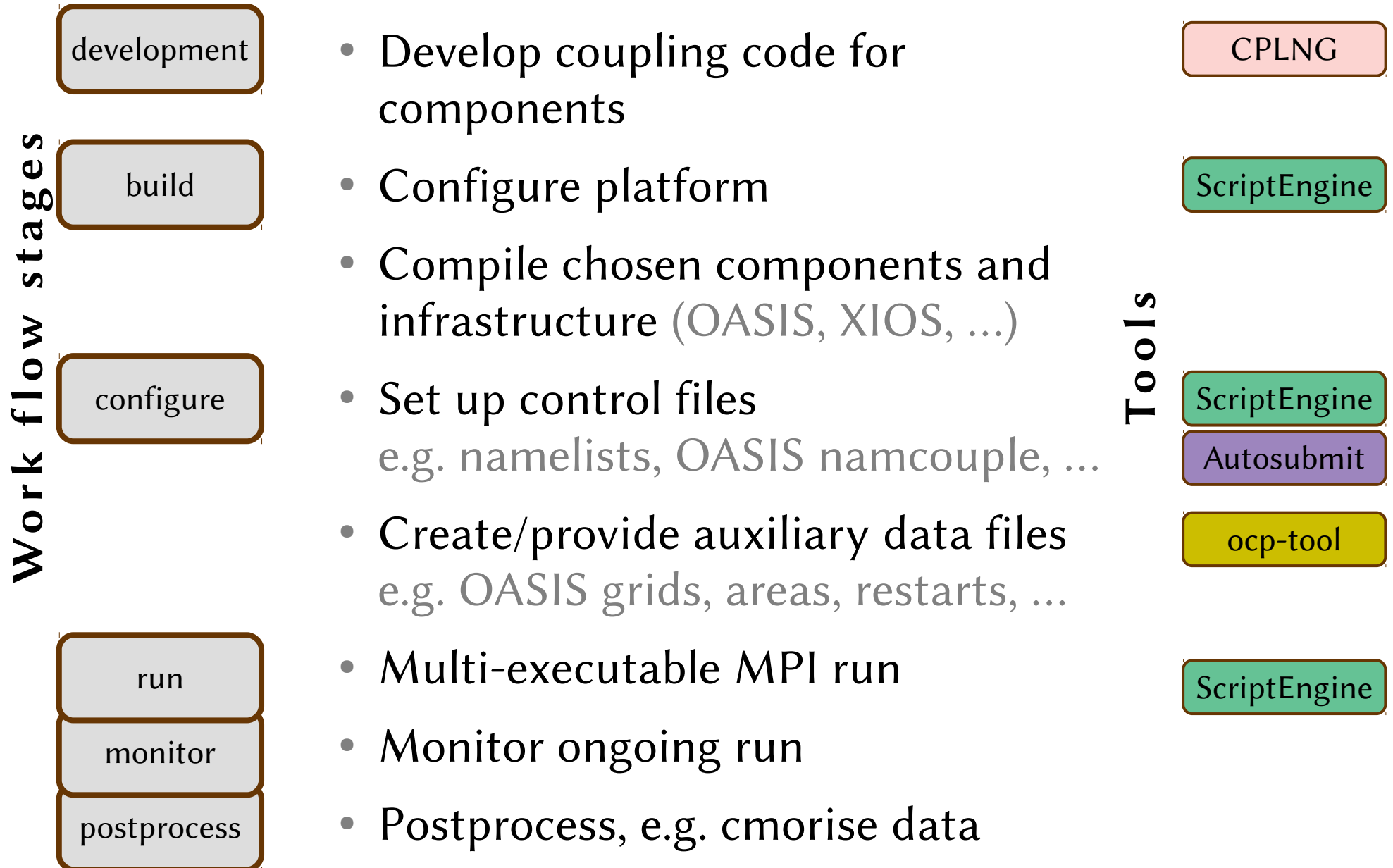
Work flow stages:



Coupling: Exchange information between components and coordinate their execution.

This is bigger than just the coupler (OASIS in our case).

Coupling everywhere



Coupling components: CPLNG

development

A generalised coupling framework for Fortran codes

High-level control of CPLNG:

```
use CPLNG          call CPLNG_CONFIG
call CPLNG_INIT    call CPLNG_FINALIZE
```

Configure a coupling field:

```
type(CPLNG_FLD) :: CPLNG_FLD(:)

CPLNG_FLD(i)%NAME      = "A_Qns_ice"
CPLNG_FLD(i)%INOUT     = OASIS_OUT
CPLNG_FLD(i)%TYPE     = CPLNG_GRIDPOINT
CPLNG_FLD(i)%STAGE    = CPLNG_STAGE_SND_OCE
CPLNG_FLD(i)%NUM_CAT  = 5
```

Send/receive fields:

```
call CPLNG_EXCHANGE(CPLNG_STAGE_SND_OCE)
```

OASIS3-MCT features

development

ECE4 upgrades OASIS3-MCT 3.0 → 4.0

We want to test:

- Review conservative remappings
- Bundled coupling fields
- Online weight calculation
- MAPPING with predefined weights and \$MAPLOC
- \$NMAPDEC=decomp_wghtfile
- Restart file writing from oasis_put

ScriptEngine: YAML+Jinja2

ScriptEngine is a lightweight and extensible framework for executing scripts written in YAML.

A ScriptEngine script in YAML, with Jinja2 expressions:

```
# echo.yml:
- context:
  planet: Earth

- echo:
  msg: "Hello, {{planet}}!"
```

Run it with the ScriptEngine command:

```
> se echo.yml
2020-09-15 14:12:07 INFO [scriptengine]: Logging configured
2020-09-15 14:12:07 INFO [scriptengine.tasks.base.context]: planet=
2020-09-15 14:12:07 INFO [scriptengine.tasks.base.echo]: Hello, {{p
Hello, Earth!
```

Configure components for build

build

Configure platform setting:

```
# platforms/nsc-tetralith.yml
- context:
  netcdf:
    base_dir: /software/netcdf/4.4.1.1/HDF5-1.8.19
    inc_dir: !noparse "{{build.libs.netcdf.base_dir}}/include"
    lib_dir: !noparse "{{build.libs.netcdf.base_dir}}/lib"
    libs: [netcdf, netcdf]
```

... and NEMO build options:

```
# templates/nemo/arch-ecearth.fcm.j2
%NCDF_INC      {% if build.libs.netcdf.inc_dir %} -I{{build.libs.n
                {% if build.libs.hdf5.inc_dir %} -I{{build.libs.hd
%NCDF_LIB      {% if build.libs.netcdf.lib_dir %} -L{{build.libs.n
                {% for lib in build.libs.netcdf and build.libs.net
```

Create NEMO build script with ScriptEngine:

```
# compile-nemo.yml
- template:
  src: nemo/arch-ecearth.fcm.j2
  dst: "{{main.src_dir}}/nemo-4.0.1/arch/arch-ecearth.fcm"
```

Compile chosen components

build

Compile all GCM components with ScriptEngine:

```
> se user-settings.yml \  
    platform/nsc-tetralith.yml \  
    compile-oasis.yml \  
    compile-xios.yml \  
    compile-nemo.yml \  
    compile-oifs.yml
```


Run config: atm/oce grids

configure

Problem: When changing the atm/oce grid configuration

- OASIS need correct grids, masks, areas
- atm/oce land-sea masks change
- Runoff basins and arrival points change

Solution:

- Past: provide files for all combinations
- **OCP-Tool** (github.com/JanStreffing/ocp-tool):
 - Developed at AWI for ECE4
 - Creates auxiliary OASIS files on the fly
 - Provides OIFS files and runoff maps automatically

Run config: OASIS namcouple

configure

Auto generate namcouple with Jinja2:

```
# namcouple.j2

[...]
$RUNTIME
    {{(24*3600*(schedule.leg.end-schedule.leg.start).days)|int}}

[...]
{% if oifs in components and nemo in components %}
# --- Oce momentum flux in U grid ---
A_TauX_oce 0_0TauX1 1 {{oasis.dt}} 2 rstas.nc EXPORTED
{{oifs.agrid}} {{nemo.ugrid}} LAG={{oasis.dt}}
P 0 P 2
LOCTRANS SCRIPR
AVERAGE
GAUSWGT D SCALAR LATITUDE 1 9 2.0
{% endif %}
```

Run the experiment

run

Config&run script for ECE4 experiment (excerpt)

```
# ece-4-gcm.yml:
- context
  main:
    experiment_id: SI11
    experiment_description: |
      Larger weights for spectral albedo mapping
    components:
      - oifs
      - nemo
      - oasis
      - xios
      - rnfm

- include:
  src: "scripts/config-{{component}}.yaml"
  ignore_not_found: yes
loop:
  with: component
  in: "{{['main'] + main.components}}"
```

Monitoring & post-processing

monitor

postprocess

- ScriptEngine can be used for both, with
 - YAML specs

```
- context
  components: [oifs, nemo, lpjg]
- do:
  command:
    name: ece2cmor
    args: "--{{component}}"
  loop:
    with: component
    in: "{{components}}"
```

- and Jinja2 templates

```
"grid":                "{{cmip6.grid}}",
{% if component == 'ifs' or component == 'lpjg' %}
  "grid_label":        "gr",
{% elif component == 'nemo' %}
  "grid_label":        "gn",
{% endif %}
"nominal_resolution":  "{{cmip6.nominal_resolution}}",
```

for configuration of components

Summary

- Coupling ESM components has various levels
 - Low-level implementation
 - Configuration management
 - Work flow management
- Information needs to travel across these levels
- A couple of tools were introduced to help with this
 - CPLNG (implementation)
 - OCP-TOOL (configuration management)
 - ScriptEngine (configuration management)