Towards a new validation system - DAVAÏ

Alexandre Mary
in coll. w/ E. Arbogast & F. Suzat

Code Training Days - Sept. 12th, 2019

Outline

Outline

Building a new cycle : process

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests
3. analyse crashes/outputs ⇒ report(s)

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests
3. analyse crashes/outputs ⇒ report(s)
4. search for crashes & causes of non-expected numerical differences

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests
3. analyse crashes/outputs ⇒ report(s)
4. search for crashes & causes of non-expected numerical differences
5. collect bugfixes → **CY##_t1.02**

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests
3. analyse crashes/outputs ⇒ report(s)
4. search for crashes & causes of non-expected numerical differences
5. collect bugfixes → **CY##_t1.02**
6. ... repeat (3-4-5) as long as necessary ...

Building a new cycle : process

1. merge N user branches (or T+R branches) → **CY##_t1.01**
2. phasers & GMAP staff : run Mitraillette tests
3. analyse crashes/outputs ⇒ report(s)
4. search for crashes & causes of non-expected numerical differences
5. collect bugfixes → **CY##_t1.02**
6. ... repeat (3-4-5) as long as necessary ...
7. when happy with results, declare **CY##_t1.nn** ⇒ **CY##T1**

Why building a new cycle is difficult

Why building a new cycle is difficult

- user branches are ***insufficiently validated***
  $\Rightarrow$ need for scientists/developers to ***systematically*** validate their branches against a wide range of model configurations and algorithms

Why building a new cycle is difficult

- user branches are ***insufficiently validated***
  $\Rightarrow$ need for scientists/developers to ***systematically*** validate their branches against a wide range of model configurations and algorithms
- much of the ***evaluation of the tests is manual*** (though `ciboulette` brought some help on that), whereas it could be automatised
  $\Rightarrow$ *save time* spent on comparing listings or fields, to have more time to dive *in the code* (looking for crashes or numerical differences)

Why building a new cycle is difficult

- user branches are ***insufficiently validated***
  $\Rightarrow$ need for scientists/developers to ***systematically*** validate their branches against a wide range of model configurations and algorithms
- much of the ***evaluation of the tests is manual*** (though `ciboulette` brought some help on that), whereas it could be automatised
  $\Rightarrow$ *save time* spent on comparing listings or fields, to have more time to dive *in the code* (looking for crashes or numerical differences)
- validation of pre-cycle is split between GMAP staff & ALADIN phasers, each with its working habits :
  $\hookrightarrow$ status of validation of a pre-cycle is spread between e-mails, different formatting and degrees of precision...
  $\Rightarrow$ need for a *clear & complete* ***dashboard***

Why building a new cycle is difficult

- user branches are ***insufficiently validated***
  $\Rightarrow$ need for scientists/developers to ***systematically*** validate their branches against a wide range of model configurations and algorithms
- much of the ***evaluation of the tests is manual*** (though `ciboulette` brought some help on that), whereas it could be automatised
  $\Rightarrow$ *save time* spent on comparing listings or fields, to have more time to dive *in the code* (looking for crashes or numerical differences)
- validation of pre-cycle is split between GMAP staff & ALADIN phasers, each with its working habits :
  $\hookrightarrow$ status of validation of a pre-cycle is spread between e-mails, different formatting and degrees of precision...
  $\Rightarrow$ need for a *clear & complete* ***dashboard***

not to mention Data Assimilation, postponed by months (when it is done)

Flaws of Mitraillette

- design not very flexible
- code duplication
- not adapted to Data Assimilation validation (dependancy between tasks, ODB handling)

Flaws of Mitraillette

- design not very flexible
- code duplication
- not adapted to Data Assimilation validation (dependancy between tasks, ODB handling)

Opportunities

- Karim leave
- ***Vortex*** : now used for **all** our operational and research configurations, and contains much ***wrapping utilities*** for DA runs
- ***OOPS*** : thanks to its object-oriented design, enables to test observation operators ***separately*** from algorithms (hence in parallel to the model)

Outline

Scripting a **Task**

1. fetch input resources for task
2. run executable
3. save output : cache/archive

Scripting a **Test**

1. fetch input resources for task
1'. *fetch reference output (for comparison)*
2. run executable
2'. *compare output to reference, and feed DAVAÏ graphical interface (database) with the comparison*
3. save output : cache/archive *(+ metadata of the task, e.g. mem/CPU)*

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)
- variational

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)
- variational
- ensemble

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)
- variational
- ensemble
- physiography

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  $\hookrightarrow$ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)
- variational
- ensemble
- physiography
- miscellaneous

Classification of the tests

- model :
  - forecast
  - fullpos
  - algorithimic components around forecast (DFI, IAU, IOserv, restart...)
  - TL & AD

  ↪ upon a wide variety of geometries (LAM/global) and model options (largely imported from *Mitraillette*)
- obs preparation
- obs operators (obstype per obstype)
- variational
- ensemble
- physiography
- miscellaneous

↪ *on small truncation/grids, for* ***efficiency*** *(a few minutes or less per test, max)*
↪ *and ± small number of obs (****memory*** *vs. representativity)*

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field
- ***Jo-Tables*** : from listings, parse Jo-Tables and compares Jo obstype by obstype

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field
- ***Jo-Tables*** : from listings, parse Jo-Tables and compares Jo obstype by obstype
- ***Fields in files*** : from FA historical or GRIB post-processed files, read fields and compares field by field (normalized bias, std and max difference)

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field
- ***Jo-Tables*** : from listings, parse Jo-Tables and compares Jo obstype by obstype
- ***Fields in files*** : from FA historical or GRIB post-processed files, read fields and compares field by field (normalized bias, std and max difference)
- ***RSS*** : from *stdout*, compares memory use

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field
- ***Jo-Tables*** : from listings, parse Jo-Tables and compares Jo obstype by obstype
- ***Fields in files*** : from FA historical or GRIB post-processed files, read fields and compares field by field (normalized bias, std and max difference)
- ***RSS*** : from *stdout*, compares memory use
- ***DrHook*** : from DrHook profiling files, compares elapse time

Comparison to reference

Each task has its specific outputs to be checked.
⇒ Attach one or several dedicated ***"experts"*** to each task, an expert being in charge of comparing two similar outputs.
So far, following experts have been developed :

- ***Norms*** : from listings, parse norms and compares numbers of different digits, step by step, field by field
- ***Jo-Tables*** : from listings, parse Jo-Tables and compares Jo obstype by obstype
- ***Fields in files*** : from FA historical or GRIB post-processed files, read fields and compares field by field (normalized bias, std and max difference)
- ***RSS*** : from *stdout*, compares memory use
- ***DrHook*** : from DrHook profiling files, compares elapse time
- ***OOPS-JoTests*** : from stdout, compares computed Jo or result of adjoint test of Jo

Ciboulaï

Ciboulaï is the name of the graphical interface :

- macro view :
  - ***all Davaï experiments from all users***
  - filtering tool among experiments
  - summary of each experiment : number and % of tests OK/KO/ ?/crashed
  - $\rightarrow$ give access to *individual experiment* : $\downarrow$

a. a task can be compared to a *"continuity"* and/or a *"consistency"* reference

Ciboulaï

Ciboulaï is the name of the graphical interface :

- macro view :
  - ***all Davaï experiments from all users***
  - filtering tool among experiments
  - summary of each experiment : number and % of tests OK/KO/ ?/crashed
  - → give access to *individual experiment* : ↓
- meso view *(individual experiment)* :
  - ***all tasks ran in the experiment***
  - filtering tool among tasks
  - summary of each task : OK/KO/ ?/crashed + RSS/elapse deltas
  - → give access to *individual task* : ↓

---

a. a task can be compared to a *"continuity"* and/or a *"consistency"* reference

Ciboulaï

Ciboulaï is the name of the graphical interface :

- macro view :
  - ***all Davaï experiments from all users***
  - filtering tool among experiments
  - summary of each experiment : number and % of tests OK/KO/ ?/crashed
  - → give access to *individual experiment* : ↓
- meso view *(individual experiment)* :
  - ***all tasks ran in the experiment***
  - filtering tool among tasks
  - summary of each task : OK/KO/ ?/crashed + RSS/elapse deltas
  - → give access to *individual task* : ↓
- micro view *(individual task)* :
  - expertise about the ***task itself*** : metadata parsed by all ***experts*** involved
  - comparison to reference(s) [a], by all ***experts***

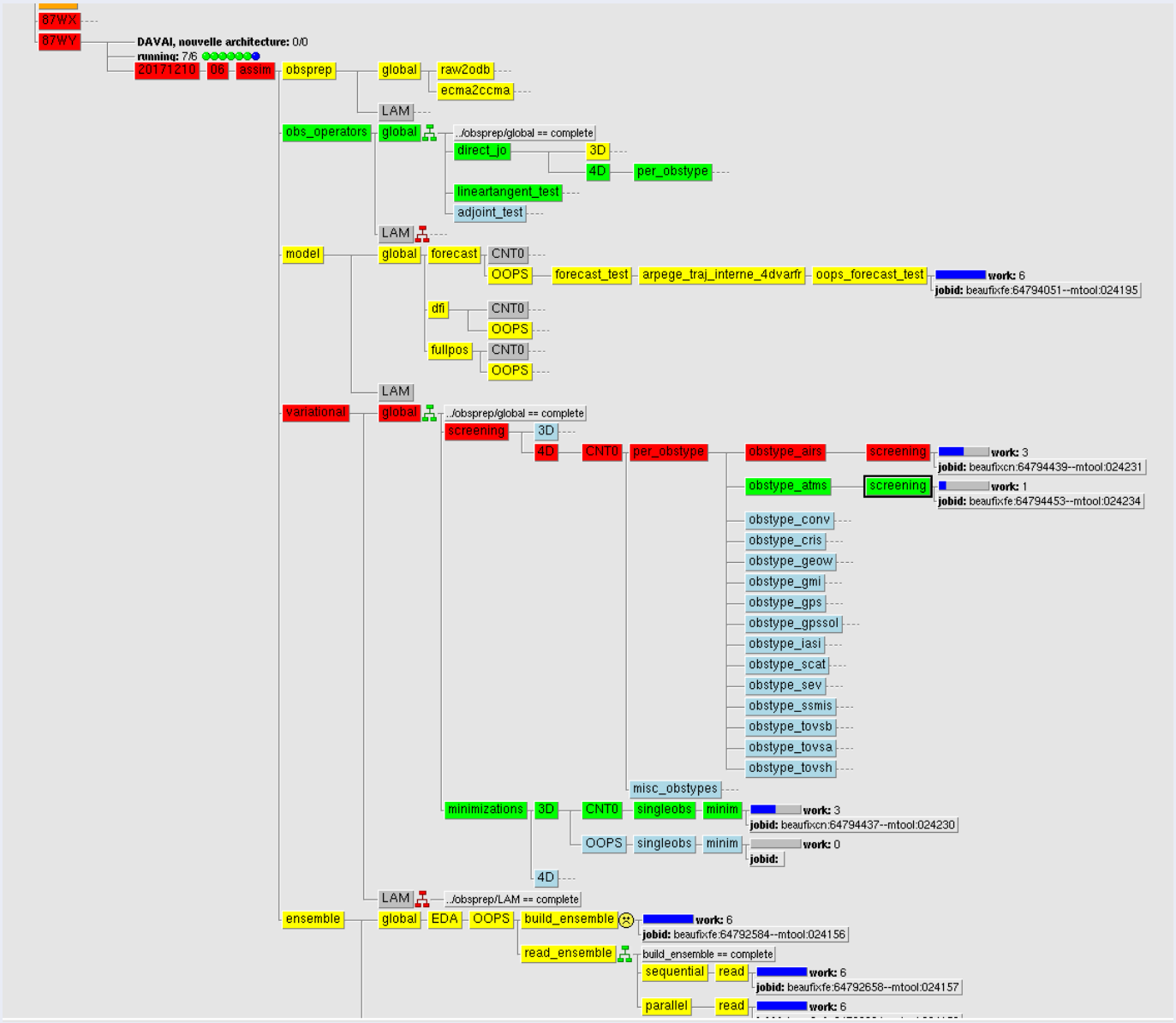
---

a. a task can be compared to a *"continuity"* and/or a *"consistency"* reference

DAVAI for/from outside MF

- portability :
  - Vortex : $\sim$ OK ?
  - Olive/Vortex works on ECMWF HPC from Météo France SMS servers
- accessibility :
  - of Olive from outside MF ?
  - export of Olive experiments/tasks to standalone Vortex scripts ?

DAVAI


87WX
87WY
DAVAI, nouvelle architecture: 0/0
running: 7/6
20171210
06
assim
obsprep
global
raw2odb
ecma2ccma
LAM
obs_operators
global
../obsprep/global == complete
direct_jo
3D
4D
per_obstype
lineartangent_test
adjoint_test
LAM
model
global
forecast
CNT0
OOPS
forecast_test
arpege_traj_interne_4dvarfr
oops_forecast_test
work: 6
jobid: beaufixfe:64794051--mtool:024195
dfi
CNT0
OOPS
fullpos
CNT0
OOPS
LAM
variational
global
../obsprep/global == complete
screening
3D
4D
CNT0
per_obstype
obstype_airs
screening
work: 3
jobid: beaufixcn:64794439--mtool:024231
obstype_atms
screening
work: 1
jobid: beaufixfe:64794453--mtool:024234
obstype_conv
obstype_cris
obstype_geow
obstype_gmi
obstype_gps
obstype_gpssol
obstype_iasi
obstype_scat
obstype_sev
obstype_ssmis
obstype_tovsb
obstype_tovsa
obstype_tovsh
misc_obstypes
minimizations
3D
CNT0
singleobs
minim
work: 3
jobid: beaufixcn:64794437--mtool:024230
OOPS
singleobs
minim
work: 0
jobid:
4D
LAM
../obsprep/LAM == complete
ensemble
global
EDA
OOPS
build_ensemble
work: 6
jobid: beaufixfe:64792584--mtool:024156
read_ensemble
build_ensemble == complete
sequential
read
work: 6
jobid: beaufixfe:64792658--mtool:024157
parallel
read
work: 6