Building new cycles: towards new practices

Alexandre Mary

ACCORD Integration Leader
Météo France

ACCORD ASW 2021
1 Building development cycles

2 Testing : Davaï

3 A bundled ecosystem?
Outline

1. Building development cycles
2. Testing: Davaï
3. A bundled ecosystem?
Contribution: timeline

<table>
<thead>
<tr>
<th>No</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Develop branch</td>
</tr>
<tr>
<td>2</td>
<td>Test, Fix issues</td>
</tr>
<tr>
<td>3</td>
<td>Push + Integration request</td>
</tr>
<tr>
<td>4</td>
<td>Review</td>
</tr>
<tr>
<td>5</td>
<td>Merge, Test, Fix</td>
</tr>
<tr>
<td>6</td>
<td>Update integration branch</td>
</tr>
</tbody>
</table>

(Substantial contributions to be announced through MG, well before starting integration)
Contribution: flow without barriers

e.g. Eoin @ met.ie

push branch

contributor

develop branch

(somewhere)

review

central repo

fetch contributor branch

merge

push integration branch

integrator

e.g. Alexandre @ MF
What will change **from a contributor perspective**

- Central repository *accessible*: from anywhere, get releases or push a contribution = `git fetch / git push` (no more manual transfers to merou or other MF internal network servers/clusters!). *(Location to be addressed)*

- **Systematical testing** of any branch, with a standardized set of Canonical configurations, as a contributor’s duty. Cf. later slides (Davaï)

- Integration requests: *web platform* (github-like)

- Reviews: more sollicitations, but avoid to discover issues later on + pre-merging
Non-Continuous Integration

(unti1 47T1)
Continuous Integration

(from 48T1 on)
Continuous Integration

- Continuous Integration (CI): 
  - validation of each branch + after each merge: branch is *continuously validated*
  - extended contribution window
- Pool of integrators around IL: 
  - shared work
  - merge conflicts solved by thematic experts
Outline

1. Building development cycles
2. Testing: Davaï
3. A bundled ecosystem?
Davaï : status

- helped to reach an *unprecedented level of validation* for common cycle CY48
- intensively used on contributions to CY48T1 : by contributors in MF, by integrator for partners
  → prevented *a number of issues*
- v1/Olive = proof of concept, but not accessible/portable outside MF, unfortunately for some contributions
  → proof of *necessity for a portable system*
- Portable v2 Davaï (Vortex, scripted definitions, Git-managed) : *under development*, prototype OK with : {build, bator, screening, minimization} for Arpege/4D & Arome/3D, full observations & by obstype
Davaï : perspectives

• Summer 2021: **first release to partners** (at least same content of tests as the v1/Olive version) + documentation

• Then some welcome help across ACCORD for:
  • pluggability of a build system other than gmkpack
  • tests of CSC Alaro and Harmonie-Arome
  • other relevant tests
  • porting to ECMWF HPC (Vortex OK)
  • set of 32b tests

• And subsequently:
  • porting to other HPCs
  • set of toy tests on workstation (GNU compiler)
  • input switch to *bundle* (separation of repositories, e.g. OOPS, SURFEX)
Outline

1. Building development cycles
2. Testing: Davaï
3. A bundled ecosystem?
Separation of interfaced projects

- Today: side-projects *forked* in the NWP repository (OOPS, SURFEX, ...)
- Tomorrow: remove them & pick from actual projects, managed on their own
  → interface: need to tie together the versions of projects that are compatible
  ⇒ *bundle*
- Maintain NWP branches therein
- Tool *ebundle* from ECMWF
- Need to adapt contribution procedure in case it addresses several projects?
Thank you for your attention.
Happy to discuss: alexandre.mary@meteo.fr
Merge of a R cycle and a T cycle

(untił now)
Merge of a R cycle and a T cycle

Convergence of EC & MF Git repositories, creation of a *common* branch

- *(the merge remains a tricky step)*
- Communications via Git commands
- More frequent and atomic updates of the merge branch
- Easier pre-phasing & post-phasing *(git blame)*
+ mutual tests
Davaï: structure

- **Input:**
  - branch
  - version of tests

- **Testing XP**
  - (directory)
  - set of jobs templates including:
    - fetch sources & build executables
    - various canonical configurations (with automatic comparison to reference)

- **Davaï (tests)**
  - (python scripts)
  - use of Vortex:
    - fetching resources
    - wrapping of executable
    - launch
    - dispatching output
  - Expertise: davaï_tbx
  - reporting results to dashboard

- **Cibouloi (dashboard)**
  - (dashboard)
  - web server
  - fed by http POSTs
  - web visualisation interface
  - reports to

- **Packages**
  - (python)

- **Davaï api**
  - (tests definitions & command-lines)
  - tasks templates
  - configuration of tasks/job (job)
  - jobs definition: dependant jobs, loops
  - launching wrappers

- **ia4h_scm**
  - (build executables)
  - Git wrapping
  - gmakepkg wrapping
  - build algorithms
  - later:
    - *bundle ?
    - *other building tools ?

- **Davaï**
  - (Vortex sub-package)
  - dedicated algorithms, resources and utilities

- **Davaï_tbx**
  - (expertise of outputs)
  - *algorithm
  - *experts parsers

- **Cibouloi**
  - (python web framework)
  - based on Django
  - HTML5 rendering

---

Building development cycles

Testing: Davaï

A bundled ecosystem?

Merci