Validations : the case of Full-Pos

- Specificities of the software
- Main processes to achieve the validations
- Survey of the « gazworks »
- Strategy of the elementary tests
- Conclusions

Ryad El Khatib – Météo-France CNRM/GMAP

November, 2002

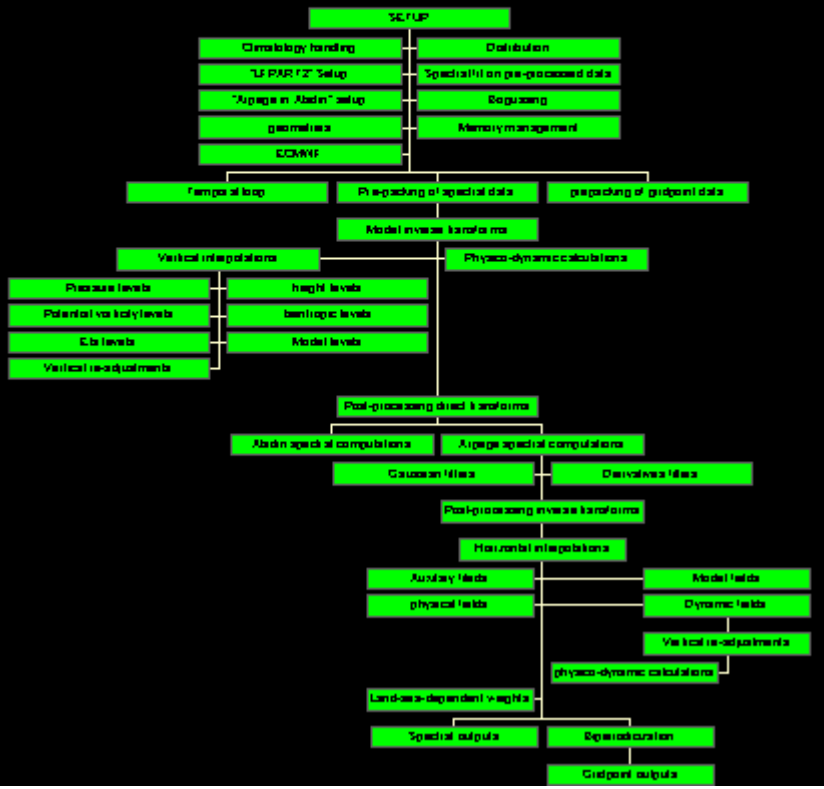
Specificities of the software

- A software embedded in the software arpege/ifs/aladin : using common parts
- A software using external packages : to be interfaced
- Phasing problems are rather limited : ECMWF misses the knowledge of the software !
- Complex interaction between arpege and aladin sides : arpege to be run with aladin software
- Strategic position of the software (4D-Var, coupling, nesting) : needs a proper validation

Main processes to achieve the validations

- Read the documentation (memorandum) : *to forsee where we are proceeding to !*
- Keep in mind the « irreversible » modifications : always problematic for the validations
- Be pragmatic :
  - The smallest change in the results must be justified
  - Take advantage of the compiler (debug, bound checking)
  - Do systematic tests (even if the job will run, for sure)
- Be methodic : to debug a complex software, do elementary tests

Survey of the « gazworks »



If you have a good knowlege
Of the software architecture and its interactions …

…You can find a way to test individually the parts of the system !

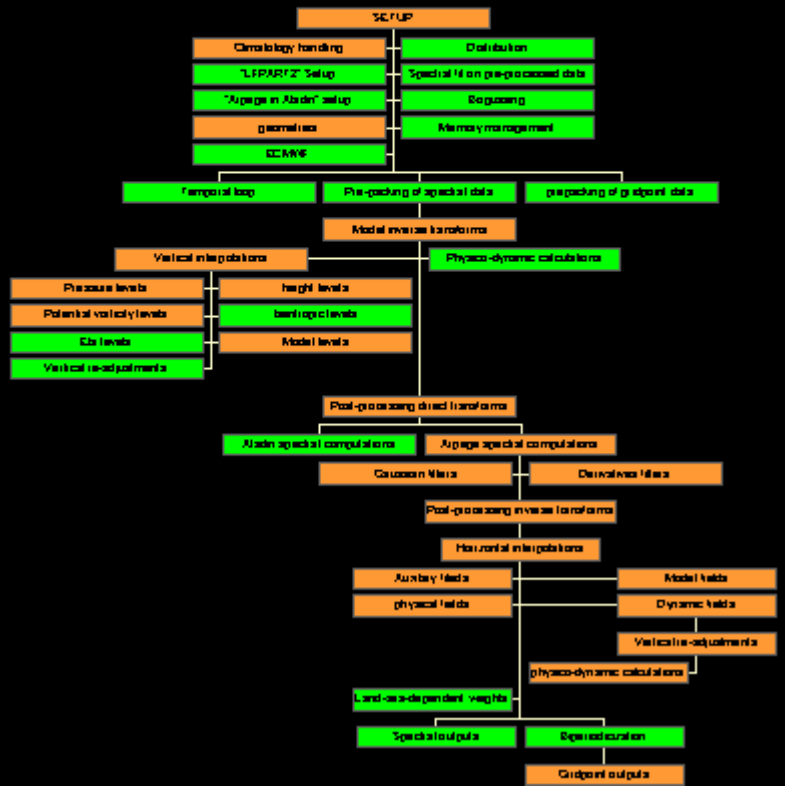
The configuration e927 « gazworks »

Active

Inactive

…Don't start to validate the software with such a configuration !!!

The Météo-France operational post-processing « gazworks »

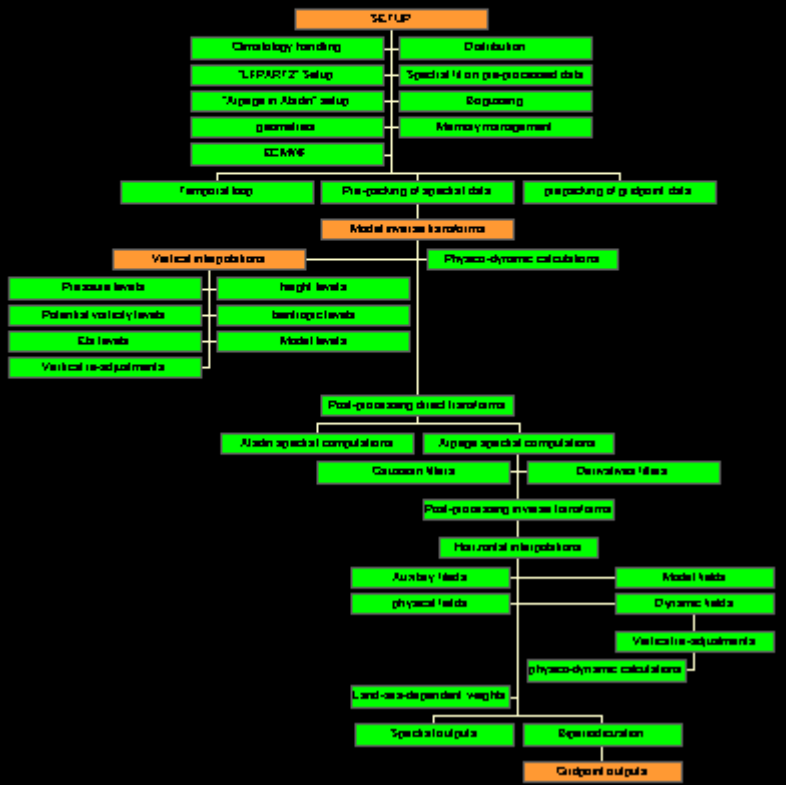


Active

Inactive

…Don't start to validate the software with such a configuration neither !!!

Strategy of the elementary tests (1)



Active

Inactive

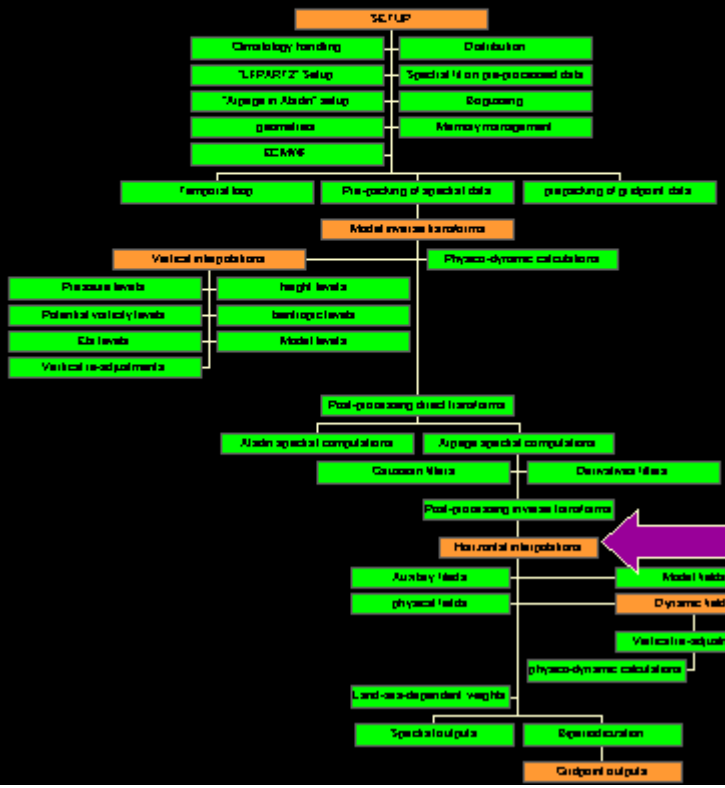
CFPFMT='MODEL',

CFP2DF='MAP_FACTOR',

NPROC=1,

…Much easier to start with !!

Strategy of the elementary tests (2)



Active

Inactive

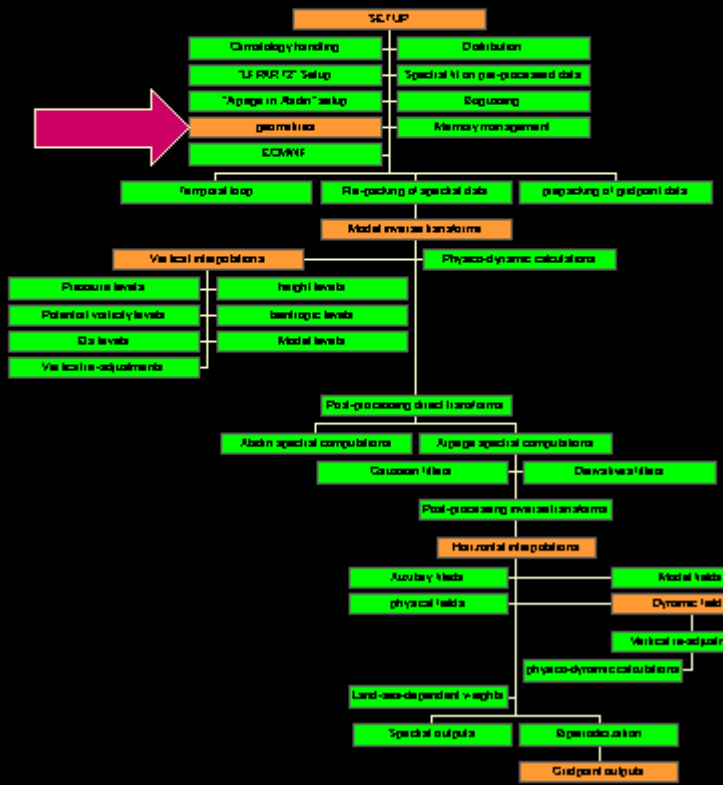
CFPFMT='GAUSS',

CFP2DF='MAP_FACTOR',

NPROC=1,

…add complexity little by little

Strategy of the elementary tests (3)



Active

Inactive

CFPFMT='LALON',

CFP2DF='MAP_FACTOR',

NPROC=1,

…add complexity little by little

Mister Ryad's sets of tests

- 32 basic tests to be submitted in hierarchical order to validate Arpege
- 30 suplementary basic tests to be submitted in hierarchical order to validate Aladin (cf « mitraillette »)
- New tests added to validate new developments
- Supplementary tests to prevent the missed bugs
- Soon : systematic tests of all the operational namelists

Mister Ryad's testing method

- Careful visual control of the listing differences between experience and reference
- Manual interventions (if possible) to exonerate an « irreversible » modification
- Plotting chart can be informative

Conclusions -Perspectives

- Documentation is essential to achieve a validation strategy
- Documentation is reading and writing
- Communication is essential : during and after debugging/validation
- Total validation is utopic ; so do as much *clever* tests as possible
- « Self-learning » tests strategy : as if you were playing chess against Kasparov
- Well-structured and modular programs make validations easier