LIBRARY ARCHITECTURE AND HISTORY OF THE TECHNICAL ASPECTS IN ARPEGE/IFS, ALADIN AND AROME IN THE CYCLE 42 OF ARPEGE/IFS.

YESSAD K. (METEO-FRANCE/CNRM/GMAP/ALGO)

July 7, 2015

Abstract:

This documentation has the aim to give a general overview of the library architecture ARPEGE/IFS, ALADIN and AROME models. Information will be given about an historical overview of the most important code rewritings, and about the dependencies between the different sub-projects.

Résumé:

Cette documentation a pour but de donner un aperçu géneral de l'architecture des librairies contenant les codes d'ARPEGE/IFS, ALADIN et AROME. On fera un rappel historique des principales réécritures de code qui sont intervenues dans le code, et on fera une description des dépendances entre les différents sous-projets.

Contents

1 Introduction and general purpose of this documentation.

The ARPEGE/IFS project has started in 1987 at METEO-FRANCE (for ARPEGE) and ECMWF (for IFS), and during its history the code has included more and more different applications, including climatic applications and limited area modelling (ALADIN, ALARO, AROME). More recently the consortium HIRLAM joined us and there are now also some HIRLAM applications in the code.

The ARPEGE/IFS software now reaches more than 25 years of existence, and during all these years the volume of code has considerably increased. The code architecture is now very complex and is becoming very difficult to learn for newcomers. More and more documentations have been written about it. This one describes the general architecture of the library containing the code, i.e. the division into projects (what do they contain?), the division of the projects into subdirectories (what do they contain?), the dependencies between these projects (what subset of routines is allowed to be called from another project, what subset of projects should remain black boxes?).

Another point we want to focus on is the history of the main evolutions of the code. We don't want to recall the main changes of the operational configurations (there are specific documentations for that) but rather give an historical summary of the main evolutions of the code architecture (data flow, externalisations, adaptations for new machines for example): that can help newcomers to understand why the architecture is currently as it is. We also try to give information about recent evolutions of the code size.

2 Library architecture.

2.1 Projects.

All the code is stored in a GIT library (PERFORCE at ECMWF). The reader entering GIT can see a first level list of "projects" under the root /home/marp001/dev.
A first group of projects are the ones which are involved in all configurations, even in those doing only forecast.

- ALADIN: specific LAM routines (limited area models, in particular ALADIN and AROME).
- ARPIFS: global model routines (ARPEGE, IFS), and routines common to global and LAM models. This is the core of the ARPEGE/IFS software.
- TRANS: spectral transforms for global models (in practical ARPEGE and IFS).
- ETRANS: spectral transforms for LAM models (in practical for ALADIN, AROME, HIRALD, ALARO).
- ALGOR: linear algebra (including minimizers, Lanczos algorithm).
- IFSAUX: some application routines, for example reading or writing on "LFI" or ARPEGE files, interface routines for distributed memory environment.
- SURF: ECMWF surface scheme.
- BIPER: bi-periodicisation package (limited area models).
- COUPLING: lateral coupling and spectral nudging for LAM models.
- MPA: upper air MESO-NH/AROME physics (also used in ARPEGE/ALADIN).
- MSE: surface processes in MESO-NH/AROME (interface for SURFEX).
- SURFEX: surface processes in MESO-NH/AROME.

A second group of projects are the ones which are used in an assimilation cycle, but not in a simple forecast:

- AEOLUS: package for pre-processing satellite lidar wind data.
- BLACKLIST: package for blacklisting.
- COPE: package for "cope" (new way to pre-process observations).
- OBSTAT: statistics of observation feedback data (only used at ECMWF).
- ODB: ODB (Observational DataBase software), needed by ARPEGE/ALADIN for their analysis or their assimilation cycle.
- SATRAD: satellite data handling package, needed to run the model analysis/assimilation.
- SCAT: scatterometers handling.

We can add the following projects:

- UTILITIES: utilities package, containing various tools like the operational FA-to-GRIB converter (PROGRID).
- SCRIPTS: scripts used at ECMWF to launch different configurations of IFS; some of them are used in the ECMWF procedures "prep_an" and "prep_ifs".
- MITRAILLE: to store "MITRAILLETTE" environment (environment to simple code validations).

Projects may have different names between ECMWF and the different libraries used at METEO-FRANCE; but the rule is to progressively go towards name harmonisation.

Standard name	former CLEARCASE (MF)	GIT (MF)	PERFORCE (ECMWF)
AEOLUS	AEO	AEOLUS	AEOLUS
ALADIN	ALD	ALADIN	(not used)
ARPIFS	ARP	ARPIFS	IFS
BIPER	BIP	BIPER	(not used)
BLACKLIST	BLA	BLACKLIST	BLACKLIST or BL?
COPE	COPE	COPE	COPE
COUPLING	COUPLING	COUPLING	(not used)
INTERPOL	(not yet implemented)		
MOCAGE	MOC	MOCAGE	(not used)
MPA	MPA	MPA	(not used)
MSE	MSE	MSE	(not used)
OBSTAT	OBT	OBSTAT	OBSTAT
ODB	ODB	ODB	ODB
SATRAD	SAT	SATRAD	SATRAD
SCRIPTS	SCR	SCRIPTS	SCRIPTS
SCAT	SCT	SCAT	SCAT
SURF	SUR	SURF	SURF
SURFEX	SURFEX	SURFEX	(not used)
ETRANS	TAL	ETRANS	(not used)
TRANS	TFL	TRANS	TRANS
UTILITIES	UTI	UTILITIES	(not used)
ALGOR	XLA	ALGOR	ALGOR
IFSAUX	XRD	IFSAUX	IFSAUX
MITRAILLE	(not used)	MITRAILLE	(not used)
OOPS	(not used)	OOPS	OOPS

2.2 More description for each project.

∗ **ARPIFS:** It contains the following directories:

- adiab:
  - Adiabatic dynamics.
  - Adiabatic diagnostics and intermediate quantities calculation, for example the geopotential height (routines GP... or GNH...).
  - Eulerian advections.
  - Semi-Lagrangian advection and interpolators (routines LA...).
  - Semi-implicit scheme and linear terms calculation (routines SI..., SP..SI..).
  - Horizontal diffusion (routines SP..HOR..).
- ald_inc:
  - function: functions used only in LAM models.
  - namelist: namelists read by LAM models.
- c9xx: specific configurations 901 to 999 routines (mainly configuration 923). Routines INCLI.. are used in configuration 923. Routines INTER... are interpolators used in configurations 923, 931, 932.
- canari: routines used in the CANARI optimal interpolation. Their names generally starts by CA.
- chem: chemistry used in ECMWF physics.
- climate: some specific ARPEGE-CLIMAT routines.
- common: often contains includes.
- control: control routines. Contains in particular STEPO and CNT... routines.
- dfi: routines used in the DFI (digital filter initialisation) algorithm.
- dia: diagnostics other than FULL-POS. One finds some setup SU... routines specific to some diagnostics and some WR... routines doing file writing.
- fullpos: FULL-POS software.
- function: functions (in includes). The qa....h functions are used in CANARI, the fc....h functions are used in a large panel of topics.
- gbrad: assimilation of radar rain observations.
- interpol: routines involved in interpolators (and halo management).
- io_serv: routines involved in some routines writing data on ARPEGE files, when switching on some optimisations.
- kalman: Kalman filter.
- module: all the types of module (variables declarations, type definition, active code).
- mwave: micro-wave observations (SSM/I) treatment.
- namelist: all namelists.
- nemo: ocean coupler used at ECMWF.
- obs_error: treatment of the observation errors in the assimilation.
- obs_preproc: observation pre-processing (some of them are called in the screening).
- ocean: oceanic coupling, for climatic applications.
- onedvar: 1D-VAR assimilation scheme used at ECMWF.
- oops: for oops layer.
- op_obs: observation horizontal and vertical interpolator.
- parallel: parallel environment, communications between processors.
- phys_dmn: physics parameterizations used at METEO-FRANCE, and HIRLAM physics, ALARO physics.
- phys_ec: ECMWF physics. Some of these routines (FMR radiation scheme, Lopez convection scheme) are now also used in the METEO-FRANCE physics.
- phys_radi: some ECMWF radiation physics routines.
- pp_obs: vertical interpolator common to FULL-POS and the observation interpolator; some of these routines may be used elsewhere.
- programs: main programs.

- raingg: rain gauge assimilation.
- sekf: simplified extended Kalman filter routines (configuration 302).
- setup: setup routines not linked with a very specific domain. More specific setup routines are spread among some other subdirectories.
- sinvect: singular vectors calculation (configuration 601).
- smos: microwave radiances.
- transform: hat routines for spectral transforms.
- utility: miscellaneous utilities, array deallocation routines.
- var: routines involved in the 3DVAR and 4DVAR assimilation, some minimizers (CONGRAD), some specific 3DVAR and 4DVAR setup routines.

∗ **ALADIN:** It contains the following directories:

- adiab: adiabatic dynamics.
- blending: blending scheme (currently only contains the procedure blend.ksh).
- c9xx: specific configurations E901 to E999 routines (mainly configuration E923). Routines EINCLI.. are used in configuration E923. Routines EINTER... are interpolators used in configurations E923, E931, E932.
- control: control routines.
- coupling: lateral coupling by external lateral boundary conditions, spectral nudging.
- dia: diagnostics other than FULL-POS.
- fullpos: FULL-POS routines.
- interpol: routines involved in interpolators (and halo management).
- module: modules used only by ALADIN routines (could be in arpifs/module too).
- obs_preproc: observation pre-processing (some of them are called in the screening).
- parallel: parallel environment, communications between processors.
- programs: main programs.
- setup: setup routines not linked with a very specific domain. More specific setup routines are spread among some other subdirectories.
- sinvect: singular vectors calculation (configuration E601).
- transform: hat routines for spectral transforms.
- utility: miscellaneous utilities, array deallocation routines.
- var: routines involved in the 3DVAR and 4DVAR assimilation, some specific 3DVAR and 4DVAR setup routines.
- wavelet: routines involved in the wavelet-Jb.

∗ **TRANS:** It contains the following directories:

- build: contains procedures.
- external: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- module: all the types of module (variables declarations, type definition, active code).
  - tpm_...F90: variable declaration + type definition modules.
  - lt...._mod.F90: active code modules for Legendre transforms.
  - ft...._mod.F90: active code modules for Fourier transforms.
  - tr...._mod.F90: active code modules for transpositions.
  - su...._mod.F90: active code modules for setup.
- programs: specific entries which can be used for TFL code validation. These routines are not called elsewhere.

∗ **ETRANS:** It contains the following directories:

- external: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- module: all the types of module (variables declarations, type definition, active code).
  - tpmald_...F90: variable declaration + type definition modules.
  - elt...._mod.F90: active code modules for N-S Fourier transforms.
  - eft...._mod.F90: active code modules for E-W Fourier transforms.
  - sue...._mod.F90: active code modules for setup.
- programs: specific entries which can be used for TAL code validation. These routines are not called elsewhere.

∗ **ALGOR:** It contains the following directories:

- external: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- internal: routines which can be called only by another ALGOR routine.
- module: all the types of module (variables declarations, type definition, active code).

Directories "external" and "internal" contain the following subdirectories:

- fourier: Fourier transforms routines.
- lanczos: Lanczos algorithm routines.
- linalg: linear algebra routines.
- minim: minimizers.

∗ **IFSAUX:** It contains the following directories:

- bufr_io: BUFR format files reading and writing.
- build.
- cma: CMA format files reading and writing.
- ddh: DDH diagnostics.
- eclite: routines coming from an old ECLIB package.
- fa: ARPEGE (FA) files reading and writing.
- fi_libc.
- fi_pthread.
- grib_mf: METEO-FRANCE GRIB format files reading and writing.
- hack.
- include: not automatically generated interfaces.
- lfi and lfi_alt: LFI format files reading and writing.
- linux.
- misc: miscellaneous decks.
- module: all the types of module (variables declarations, type definition, active code). There are a lot of mpl...F90 modules for parallel environment (interface to MPI parallel environment).
- parallel: parallel environment, communications between processors.
- programs: main programs.
- support: miscellaneous routines.
- svipc: contains only svipc.c .
- utilities: miscellaneous utilities.

∗ **BIPER:** It contains the following directories:

- build: contains procedures.
- external: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- module: all the types of module (variables declarations, type definition, active code).
- programs: specific entries which can be used for BIPER code validation.

∗ **COUPLING:** It contains the following directories:

- external: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- module: all the types of module (variables declarations, type definition, active code).
- programs: specific entries which can be used for COUPLING code validation.

∗ **SURF:** It contains the following directories:

- build: contains procedures.
- external: routines which can be called from another project.
- function: specific functions.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- module: all the types of module (variables declarations, type definition, active code).
  - yos_...F90: variable declaration + type definition modules.
  - su...._mod.F90 but not surf...._mod.F90: active code modules for setup.
  - surf...._mod.F90, v...._mod.F90: other active code modules.
- make.
- offline.

∗ **MPA:** It contains a first level of directories:

- dummy: empty versions of some routines.
- chem: chemistry.
- conv: convection.
- micro: micro-physics.
- turb: turbulence.
- programs: main programs.

Each directory contains the following subdirectories:

- externals: routines which can be called from another project.
- include: all the "include" decks (functions, COMMON blocks, parameters).
- interface: not automatically generated interfaces which match with the "external" directory routines.
- internals: other non-module routines; they cannot be called from another project.
- module: all types of modules.

∗ **MSE:** It contains the following directories:

- dummy: empty versions of some routines.
- externals: routines which can be called from another project.
- interface: not automatically generated interfaces which match with the "external" directory routines.
- internals: other non-module routines; they cannot be called from another project.
- module: all types of modules.
- new.
- programs: main programs.

∗ **SURFEX:** It contains the following directories (with sometimes an additional subdivision):

- ASSIM.
- OFFLIN: some IO routines.
- TOPD.
- TRIP: rivers and flooding model.
- SURFEX: all other routines.

∗ **AEOLUS:** Not detailed: contains a lot of directories, routines and procedures.

∗ **BLACKLIST:** It contains the following directories:

- compiler.
- include: not automatically generated interfaces, functions, and some other includes.
- library: the only containing .F90 decks.
- programs: main programs.
- scripts.

∗ **COPE:** Not detailed: contains C++ routines and procedures.

∗ **OBSTAT:** It contains the following directories:

- bias_sat.
- data.
- doc.
- examples.
- module.
- satmon.
- src.

∗ **ODB:** Not described in detail; for more information see the specific documentations (IDODBO), (IDODBT), (IDODBU), (IDODBZ). This project has its own entries.

∗ **SATRAD:** It contains the following directories:

- bias.
- cmem.
- emiss.
- include.
- interface.
- module.
- mwave.
- onedvar.
- pre_screen.
- programs: main programs.
- rtlimb.
- rttov.
- satim.

Not described in detail; more information has to be provided by someone who knows the content of this project, but there is currently no specific documentation about this topic.

∗ **UTILITIES:** It contains the following directories:

- aca: prepare academic file for 1D model.
- add_cloud_fields: program to add 4 cloud variables (liquid water, ice, rainfall, snow) in ARPEGE files.
- addozoaer: program to add ozone and aerosols constants in ARPEGE files.
- addsurf: programs to add fields in ARPEGE files.
- bcov_lam: diagnostic for LAM covariances files.
- combi: combination of perturbations in an ensemble forecast (PEARP).
- ctpini: routines for CTPINI applications (inversion of potential vorticity fields).
- dategrib: program to read a date on GRIB files.
- gobptout: PROGRIB? (convert ARPEGE files contained post-processed data into GRIB files).
- pearome: ensemble forecast for AROME (PEARO).
- pinuts: PINUTS applications, for example to create ALADIN domains.

- pregpssol: surface GPS processing.
- progrid: PROGRID? (convert ARPEGE files contained post-processed data into GRIB files).
- progrid_cadre: cf. progrid?
- rdc: former configuration 911 (makes dilatation/contraction matrices).
- sst_nesdis: program to read the SST on the BDAP.
- sst_netcdf.

This project has its own entries. Some of the directories have an internal subdivision (include, module, programs, src) which must later be generalised to all directories.

∗ **MITRAILLE:** It contains the following directories:

- doc: documentation (pdf).
- namelist: namelists used in MITRAILLETTE.
- procedure: main procedure.
- pro_file: PRO_FILE.. files containing configurations which must be validated.
- protojobs: scripts to be launched, and some environment files.

3 History of the code architecture, and main changes.

3.1 Introduction.

The ARPEGE/IFS project has started in 1987 at METEO-FRANCE (for ARPEGE) and ECMWF (for IFS), and has lead to an operational use in 1992 at METEO-FRANCE and in 1994 at ECMWF. For the METEO-FRANCE side, the aim was to merge the old couple EMERAUDE (global spectral model) and PERIDOT (limited area grid-point model) into one global spectral model, introducing a variable-mesh geometry allowing to have the same resolution on France as for the old model PERIDOT (in the beginning of the 1990s this resolution was roughly 30 km). At METEO-FRANCE, the first operational use of ARPEGE with a variable-mesh geometry has been performed during summer 1993. During the 1990s, climate versions of ARPEGE/IFS have been developed on both sides.

In the beginning of the 1990s, a limited area spectral version of ARPEGE (called ALADIN) has been developed, and a first operational use has been done in 1994; this model is developed in collaboration with some other countries, principally some Eastern countries, but some Mediterranean countries (Portugal, Morocco, Tunisia, and more recently Algeria) have joined the list of co-developers. An important part of the code is common to the ARPEGE/IFS and ALADIN software. The current operational versions of ALADIN have a mesh-size around or slightly below 10 km.

A non-hydrostatic limited-area model (called AROME) has been put into operations in December 2008. The adiabatic part of the model uses the ALADIN-NH code. The mesh-size is now 1.3 km.

A version of the physics, more oriented towards the range of mesh-sizes 3 to 10 km, has been developed by some of our ALADIN partners. ALARO is a version of ALADIN with this physics package.

Some HIRLAM features have entered the ARPEGE/ALADIN code (HIRLAM physics).

3.2 Historical evolution of the main architecture features, and main code rewritings.

The purpose is to give an history of the main technical changes which occurred in the code, but not to give an history of the operational changes (there are specific documentations for that).

- 1987: the project ARPEGE/IFS was launched. Some pieces of codes of the old models EMERAUDE and PERIDOT have been re-used, with some rewritings.
- 1990: ALADIN early steps. Until year 2002, the main cycles of ALADIN were not completed at the same time as those of ARPEGE/IFS, and the ALADIN cycles had their own numbering.
- 1990?: first version of the 3TL (= three-time level) semi-Lagrangian scheme.
- 1987-1992: some utilities have been imported from some other softwares, for example for assimilation purposes. The CANARI optimal interpolation assimilation software has been coded.
- 1991: the first version of the DDH (French acronym for "Diagnostics par Domaines Horizontaux") diagnostics package has been coded.
- 1991: configurations 911 and 912 entered the code.
- 1991-1992: the 3TL semi-Lagrangian code was rewritten, partly according to the semi-Lagrangian code of the old ECMWF model, in order to have a better optimized code, and also a numerically stable treatment of the continuity equation.
- 1991-1992: the EMERAUDE-PERIDOT METEO-FRANCE physics has been adapted (including the compliance with the DOCTOR norm) and interfaced with the ARPEGE dynamics. ECMWF did the same thing for its physics package (adapt the physics package of the old ECMWF model). At this period, the same physical interface (APLPAR) was used for both METEO-FRANCE and ECMWF physics.
- We also notice that at the beginning of year 1992 most of the currently existing configurations were already existing at this time, excepted maybe for configuration 131 (incremental 4D-VAR); there were some other configurations which have had a short live life (configuration 99 to validate some adiabatic grid-point calculations, configuration 151 for non incremental 3DVAR, configuration 821 for sensitivity experiments applied to a shallow-water model). We should notice that at this stage the ARPEGE/IFS code had already an adjoint code and a linear tangent code (for the Eulerian advection scheme only) and some configurations using it (801, 821 for example) for research purposes.
- 1992: there were a significant rewriting of the grid-point calculations in order to save memory. Until 1992, the grid-point arrays contained all the grid-point data (arrays dimensioned with NDLON or NDLSUR, and NDGL). The rewriting done in 1992 (at ECMWF) introduced the following main types of features:
  - The NPROMA-packets.

  - An IO scheme (for ECMWF purposes only) which has been kept a couple of years, switched on by a set of LIO... keys.
  - The two-part structure in the grid-point calculations which still remains in some routines naming: XXX/XXXLAG, for example CPG/CPGLAG, COBS/OBSHOR. This two-part structure was well designed for the multi-processors (multitasking) shared memory computers available at these times (CRAY).

  This is during this recoding that the following structure appeared: STEPO $->$ SCAN2M $->$ CPG and CPGLAG, and for the semi-Lagrangian calculations the buffers filling (LACDYN) were done under CPG, the interpolations (LAPINE, ancestor of LAPINEA+LAPINEB) were done under CPGLAG. Unlagged physics (APLPAR) was done under CPG, lagged ECMWF physics (CALLPAR) was done under CPGLAG.
- Around 1992: possibility to have a reduced Gaussian grid, and to have irregularly spaced definition of the hybrid levels (option LREGETA=F).
- September 1992: ARPEGE replaced EMERAUDE for operations (T79L15c1).
- 1993: separate physics interfaces for METEO-FRANCE (APLPAR) and ECMWF (CALLPAR).
- 1993: the first version of the non-hydrostatic code (Eulerian scheme only) was implemented in ALADIN.
- 1993?: the ancestor of the 927-configuration of FULL-POS has been coded, to provide ARPEGE initial files with a stretched/tilted geometry from another geometry: the so called 926 configuration. The same thing has been coded to provide ALADIN initial and coupling files from ARPEGE files.
- September 1993: the operational ARPEGE model at METEO-FRANCE became a tilted/stretched one (T95c3.5), but it still used the Eulerian advection scheme. PERIDOT was stopped.
- March 1994: ECMWF replaced its old model towards IFS for operational applications (with a semi-Lagrangian scheme).
- 1993 to 1995: the spectral calculations (semi-implicit scheme, horizontal diffusion) were progressively adapted to the variable mesh (LSIDG option in the semi-implicit scheme) in ARPEGE.
- May 1994: First operational version of ALADIN (ALADIN-LACE). A couple of months later (1995-1996), ALADIN-FRANCE started for METEO-FRANCE operational applications.
- Spring 1994: the code, which had been managed under SCU (on a CDC using NOS/VE as system language) until this time, has been ported to CLEARCASE (HP station "becarre" using UNIX). The cycle ported was CY11. The current shearing into subdirectories has been introduced.
- 1994 (or 1995?): a first version of the two-time level (2TL) semi-Lagrangian scheme has been coded.
- Summer 1995: SL3TL (three-times level semi-Lagrangian scheme) operational at METEO-FRANCE.
- Summer 1996: SL2TL (two-times level semi-Lagrangian scheme) operational at METEO-FRANCE.
- 1994-1996: the FULL-POS software has been coded. The purpose was to replace:
  - the old configuration 926.
  - the old vertical post-processing (containing a vertical interpolator under POS which has been re-used in FULL-POS).
  - the old external post-processing software POTIPA, which has been used at METEO-FRANCE for operations until 1996.

  FULL-POS replaced POTIPA in the operational version of ARPEGE in 1996.
- 1996: the DFI initialisation, first developed in ALADIN in the early 90s, was adapted to ARPEGE. It replaced the old NMI initialisation in the operational version of ARPEGE in 1996.
- 1995-1997: the distributed memory code has been introduced in ARPEGE/IFS, in order to be used on the FUJITSU VPP distributed-memory computers. This was a very huge task, leading to a significant rewriting of the code, and this is at this stage that some variables were duplicated into a global version (name ending by G) and a local version (name ending by L): for example the couple NDGLG/NDGLL replaced the old variable NDGL. Some routines were duplicated, with a shared memory version (name ending by SM) and a distributed memory version (name ending by DM). For example, the old routine SCAN2M was split into SCAN2MSM and SCAN2MDM.

  ECMWF switched on a VPP for operational applications in 1997.
- 199?-1997: the 3D-VAR assimilation scheme has been coded in the global model for constant and variable mesh. It became operational first at ECMWF (1996). It became operational at METEO-FRANCE in may 1997. A first version of the 4DVAR assimilation scheme has been coded for ECMWF operational applications: it became operational at ECMWF in 1997.
- October 1997, cycle 18: a huge automatic code rewriting has been done, in order to remove some dirty features, and also to make most of the code compliant with the F90 norm (the code were previously only using FORTRAN-77 features, with some FORTRAN-66 residues). This rewriting has been done on all the routines of the projects ARP and ALD and most routines of the project XRD.

- June 1998: switch of the operational METEO-FRANCE ARPEGE model (and also ALADIN-FRANCE) on the FUJITSU-VPP, use of the distributed memory code (key LMESSP=T).
- June 1998 (CY19): the old option LRPOLE=T (use poles in the Gaussian grid) has been removed from the code.
- March 1999 (CY21): a new huge automatic code rewriting has been done: switch to the F90 free format instead of the F90 fixed format.
- 1999: the tangent linear and adjoint codes of the first version of the ALADIN non-hydrostatic code (variables P_0 and d_0) has been implemented.
- 1997-2000: The 4DVAR assimilation scheme has been adapted to METEO-FRANCE purposes: incremental 4DVAR with stretched/tilted geometry in the high resolution applications (guess, screening, trajectory update, CANARI surface assimilation) and unstretched/untilted geometry in the low resolution applications (minimisations).
- June 2000: the 4DVAR assimilation scheme became operational at METEO-FRANCE.
- 2000?: migration of the CLEARCASE libraries from "becarre" to "andante".
- 2000-2002: a significant cleaning of obsolete options has been done, for example the old shared memory code, the IO scheme (which has been no longer used at ECMWF for a couple of years), configurations 203, 423, 523, etc... But the actual proportion of code which has been removed was less than 10%.
- 2000-2005: a deep rewriting of the grid-point calculations has been done, and this huge task has been spread on several years. The direct code has been first rewritten, and then the linear tangent code and the adjoint code. The main visible features of this rewriting are:
  - the replacement of the two-block structure XXX/XXXLAG by a multi-block structure. Example CPG+CPGLAG has been replaced by CPG_GP+MF_PHYS+CPG_DYN+CPG_END+CALL_SL+EC_PHYS+CPGLAG (but CPG_GP+MF_PHYS+CPG_DYN+CPG_END have been gathered in CPG in order to avoid very long routines). For example the new routine CALL_SL (managing the semi-Lagrangian interpolator, previously under CPGLAG) has been introduced in CY24 (year 2001).
  - a big cleaning of very long and messy routines (CPG): code is more modular and easier to read.
  - the introduction of the new routine GP_MODEL between STEPO and CPG+...+CPGLAG (CY25, in 2002).
  - the splitting of LAPINE into LAPINEA and LAPINEB in the semi-Lagrangian interpolator.
  - the recoding of the TL and AD codes according to their direct counterparts, often with 2 or 3 years of delay. For example the splitting of LAPINEAD into LAPINEAAD and LAPINEBAD has been completed in 2005 only (CY30).
- 2002-2005: additionally to the previous grid-point rewritings, there were new options and significant rewritings in the non-hydrostatic code.
  - 2002, 2003 (CY25, CY26): Introduction of new formulations.
  - 2004 (CY29): removal of obsolete options (for example LPC_NOTR, LFULLIMP, NVDVAR=0,1 or 2, NPDVAR=0 or 1).
  - 2004, 2005 (CY29, CY30): The non-hydrostatic code, originally coded only in ALADIN, has been extended to ARPEGE. A complete rewriting has been done to make the code easier to read and more efficient (including the removal of the old routine GNHPDVD).
  - 2006 (CY31): additional optimisations have been done in the d_4 option of the non-hydrostatic code (removal of the old auxiliary variable, replacement by an additional thermodynamic one). The tangent linear and adjoint codes of the first version of the ALADIN non-hydrostatic code have been removed.
- 2001 (CY24): spectral transforms, which were previously in the projects ARP and ALD, have been externalised: new project TFL for ARPEGE, TAL for ALADIN.
- 2001 (CY24): new project ODB, for observations files, replacing the old format CMA. Replacement of the old format CMA by the format ODB in the operational version of ARPEGE.
- 2002 (CY25): new projects BLA, OBS, SAT, SCR.
- 2002 (CY25): coding deep layer atmosphere equations (Bromley and White) in the hydrostatic model (key LVERCOR=T).
- March 2003 (CY26): from this cycle ARP and ALD have the same numbering and are released at the same time.
- Summer 2003: the stretching coefficient has been reduced from 3.5 to 2.4 in ARPEGE-METROPOLE.
- Summer-autumn 2003 (CY27): new GMV and GFL data flow for grid-point calculations.
- November 2003 (CY28): a new huge automatic code cleaning has been done: declaration of dummy argument variables with INTENT attribute, removal of dirty features, introduction of automatic interfaces.

- End of 2003 (between CY28 and CY29): externalisation of the ECMWF surface scheme in a new project SUR.
- Summer 2004: Deliverance of the software OLIVE at METEO-FRANCE (to launch experiments close to operational configurations).
- December 2004 (CY29): from this cycle all projects are gathered in one CLEARCASE library only (previously there were one library per project).
- 1997-2004: developments have been done to adapt the 3DVAR to ALADIN.
- July 2005: the 3DVAR assimilation became operational in ALADIN-FRANCE.
- September 2005 (CY30): new projects MPA and MSE (code coming from MESO-NH) for the AROME physics. First prototype of AROME.
- 2005-2006 (CY30, CY31): harmonisation of spectral calculations between ARPEGE and ALADIN. Non-hydrostatic model spectral calculations have been ported on ARPEGE; ESPC has been split into ESPCSI, ESPNHSI and ESPCHOR in ALADIN; in ESPCSI the algorithm has been optimised into a one using calculations in the normal modes space (removal of the old array SIEHEL).
- 2006 - early 2007: adaptations of the ARPEGE/IFS code to the NEC architecture.
- First quarter of 2006 (CY31): first HIRALD prototype (introduction of HIRLAM physics); early stages of the collaboration HIRLAM-ALADIN.
- First quarter of 2006 (CY31): TL and AD codes of the semi-Lagrangian scheme adapted to the stretched/tilted version of ARPEGE.
- Summer-autumn 2006 (CY32): new data flow for the surface fields.
- Autumn 2006 (CY32): prototype ALARO0 (version 0 of the ALARO physics, designed for horizontal mesh-sizes between 3 and 10 km).
- Autumn 2006 (CY32): TL code of the semi-Lagrangian scheme adapted to ALADIN.
- Spring 2007 (CY32T2): AD code of the semi-Lagrangian scheme adapted to ALADIN.
- April 2007 (CY32T0+CY32T1): migration of the CLEARCASE libraries from "andante" to "merou"; OBS renamed into OBT.
- May 2007: the operational models at METEO-FRANCE now run on a NEC-SX8 computer instead on a FUJITSU VPP one.
- October 2007 (CY33): new project AEO (AEOLUS).
- October 2007 (CY33): new configuration 302 (simplified extended Kalman filter).
- April 2008 (CY33T1): new project BIP (bi-periodicisation).
- May 2008 (CY34): evolutions in the NH model:
  - VFE for NH-PDVD model (i.e using pressure departure variable and vertical divergence).
  - Prototype of NH-GEOGW model (NH model using Φ and gw prognostic variables, VFE only).
  - Wood and Staniforth formulation of NH deep-layer equations (NH-PDVD model, partly implemented).
- August 2008 (CY35): new project XLA (linear algebra, minimizers, Lanczos algorithm), significant changes in library structure, significant amount of routine renaming.
- October 2008 (CY35T1): removal of configuration 951 and of some obsolete options (for example enhanced diffusion).
- December 2008: AROME has been put into operations.
- February 2009 (CY35T2): significant rewritings in the semi-Lagrangian scheme (SLHD, dataflow, interpolators).
- 2009-2010: merge of routines doing roughly the same thing (for example: different versions of SLEXTPOL, SLCOMM, DISGRID, DIWRGRID, DIWRSPEC, TRIDIA, MORTHO, calculation of Gaussian weights, SLRSET).
- September 2009: the operational models at METEO-FRANCE now run on a NEC-SX9 computer.
- November 2009 (CY36T1): removal of configuration 912; externalisation of configuration 911 in UTI.
- November 2009 (CY36T1): new project SURFEX.
- Spring 2010 (CY36T2): removal of configurations 903 and 940.
- 2010 (CY37): significant cleaning (removal of useless dummy arguments, proper intent of the other ones, removal of useless declarations, and more generally removal of some features generating norm violations).
- 2010 (CY37): significant rewritings in ODB.
- 2011-2012 (CY39): rewriting of LAM coupling and spectral nudging (new project COUPLING); externalisation of relaxation and temporal interpolation.

- 2012 (CY39): rewriting of GOMS dataflow and observation interpolator.
- 2012 (CY39): removal of NMI (normal mode initialisation) code.
- 2012 (CY39): possibility to use fast Legendre transforms.
- 2012-2013 (CY39 and CY40): complete rewriting of FULL-POS for more flexibility (new routine STEPO_FPOS, new 928-configuration).
- 2012-2013: porting code to scalar machines; in particular significant developments have been done to optimize IO (new software io_serv, optimisation of ARPEGE and LFI files IO).
- 2013 (CY40): significant rewritings in the ECMWF physics interface (number of arguments and size of some routines, like CALLPAR, have considerably decreased).
- 2013 (CY40): some include files (.h files) have been renamed; appendix is now .func.h for functions, .nam.h for namelists.
- 2011-2016?: more object-oriented dataflow (OOPS project). That includes for example encapsulation of model dataflow and geometry object.
- 2013-????: complete rewriting of observation pre-processing (new project "COPE" in CY40).
- Summer 2013: migration of libraries from CLEARCASE to GIT.
- January 2014: switch of METEO-FRANCE operational suites on BULL machines.
- March 2014 (CY40T1): removal of command line options, replaced by namelist NAMARG.
- 2014 (CY41 and CY41T1): removal of the former FULL-POS 927 and 928.
- 2014: possibility to use the IO server.
- 2015 (CY41T1): new project MITRAILLE to store "MITRAILLETTE" environment.

3.3 Date of deliverance of the main cycles.

- CY00: 1987 or 1988?
- CY01, CY02: 1988, 1989?
- CY03: early 1990?
- CY04: spring 1990.
- CY05: autumn 1990.
- CY06: spring 1991.
- CY07: autumn 1991.
- CY08: spring 1992.
- CY09: autumn 1992?.
- CY10: mid 1993?.
- CY11: early 1994 (first cycle under CLEARCASE).
- CY12: autumn 1994.
- CY13: April 1995.
- CY14: September 1995.
- CY15: March 1996.
- CY16: February 1997.
- CY17: April 1997.
- CY18: October 1997.
- CY19: June 1998.
- CY20 and CY21: March 1999.
- CY22: December 1999.
- CY23: July 2000.
- CY24: February 2001.
- CY25: February 2002.
- CY26: March 2003.
- CY27 and CY28: November 2003.
- CY29: December 2004.
- CY30: September 2005.
- CY31: May 2006.

- CY32: December 2006.
- CY33: December 2007.
- CY34: August 2008.
- CY35: September 2008.
- CY36: July 2009.
- CY37: November 2010.
- CY38: November 2011.
- CY39: November 2012.
- CY40: July 2013 (last cycle on CLEARCASE, first cycle on GIT).
- CY41: July 2014.
- CY42: June 2015.

The exact chronology of cycles before CY11 (1994) is difficult to recover because no trace of these cycles has been kept and no trace of the chronology has been kept in the still accessible documentations.

3.4 Versions of SURFEX matching with development cycles.

- Surfex v5.1: CY36T1 and CY37.
- Surfex v6.0 (+ some optimisations): CY37T1 and CY38.
- Surfex v7.2: CY38T1, CY39, CY39T1 and CY40.
- Surfex v7.3: CY40T1, CY41, CY41T1 and CY42.

4 Evolution of the code size.

The evolution of the code size between 1995 and 2010 is described in CY38 version of this documentation and is not recalled there (annual increase during this period was around 6%). To sum-up:

- Number of decks in project ARPIFS has increased from 1949 decks in CY14 (sept 1995) to 3656 decks in CY37 (nov 2010).
- Total number of decks has increased from 4393 decks in CY19 (june 1998) to 13228 decks in CY37 (nov 2010).
- Number of lines in project ARPIFS has increased from 1088015 lines in CY32 (dec 2006) to 1653090 lines in CY37 (nov 2010).
- Total number of lines has increased from 2394706 lines in CY32 (dec 2006) to 5037834 lines in CY37 (nov 2010).

* Number of decks and lines (in parentheses): evolution between 2011 and 2015:

	CY38 NOVE 2011	CY39 NOVE 2012	CY40 JULY 2013	CY41 JULY 2014	CY42 JUNE 2015
arpifs	3698 (1061192)	3679 (1059480)	3795 (1078735)	3836 (1106039)	3903 (1405620)
aladin	228 (59712)	213 (56288)	212 (55725)	214 (56836)	217 (57699)
etrans	97 (15069)	99 (15389)	97 (15478)	100 (15712)	100 (15742)
trans	145 (21858)	158 (24053)	154 (24439)	157 (24677)	158 (27654)
ifsaux	578 (108002)	594 (109548)	467 (122557)	528 (131525)	584 (140956)
algor	124 (21276)	126 (22439)	130 (23255)	131 (24544)	131 (25247)
biper	16 (3057)	18 (3402)	18 (3402)	18 (3501)	18 (3501)
coupling	/	16 (1903)	17 (2203)	17 (2140)	17 (2140)
mpa	564 (153725)	570 (155510)	574 (156558)	578 (158831)	582 (159723)
mse	236 (19341)	262 (24509)	283 (28041)	283 (29352)	320 (31866)
surfex	1229 (336606)	1351 (327026)	1350 (327382)	1525 (477028)	1531 (480751)
blacklist	44 (11277)	43 (11270)	43 (11457)	44 (11588)	44 (11606)
cope	/	/	221 (14369)	221 (14369)	221 (14369)
aeolus	923 (1559301)	1001 (2656807)	633 (317368)	633 (317368)	651 (327637)
obstat	220 (137187)	222 (136793)	222 (137228)	224 (139481)	224 (139540)
odb	3538 (616254)	3529 (566870)	3302 (492139)	3318 (497635)	3359 (467365)
satrad	666 (165805)	672 (180074)	686 (184138)	741 (196293)	748 (197882)
scripts	1190 (225193)	1190 (225193)	1190 (225193)	1532 (292645)	1532 (292645)
surf	139 (44619)	145 (45989)	145 (46218)	336 (85245)	306 (85508)
utilities	100 (25270)	107 (30852)	111 (32304)	147 (33707)	152 (36201)
mitraille	/	/	/	/	241 (90256)
oops	/	/	/	/	244 (26342)
scat	57 (11680)	58 (11797)	58 (11798)	58 (11797)	69 (12555)
arpifs to coupling	4886 (1290166)	4903 (1292502)	4890 (1325794)	5001 (1364974)	5128 (1678559)
mpa+mse+surfex	2029 (509672)	2183 (507045)	2207 (511981)	2386 (665211)	2433 (672340)
blacklist to scat	6877 (2796585)	6967 (3865645)	6611 (1472212)	7254 (1600128)	7791 (1701906)
total	13792 (4596423)	14053 (5665192)	13708 (3309987)	14641 (3630313)	15352 (4052805)

5 Coding norms.

The reader can refer to documentation (IDCS11) to have an idea of the coding norms, and all people working on the ARPEGE/IFS environment should read (IDCS11), because there are currently too many code features which are not norm-compliant.

We can add some remarks:

- Most routines are written in free format F90, and some projects use also some C-routines. There are some projects containing a couple of imported routines still written in F77 or in F90 fixed format. Projects ARPIFS, ALADIN, TRANS, ETRANS, SURF only contain F90 free format routines. It is expected to introduce C++ routines in ARPIFS (upper levels of call-tree only).
- DOCTOR norm: we should notice that the MSE, MPA and SURFEX projects have a specific DOCTOR norm (the MESO-NH DOCTOR norm) which is often different from the ARPEGE DOCTOR norm which is used for example in the project ARPIFS. More generally some imported routines (projects IFSAUX, ALGOR, OBSTAT, ODB, SATRAD, UTILITIES) are not always DOCTOR-compliant.
- About variable naming, prefix and appendix must follow norms. Root name must be meaningful in English, and consistent throughout the code; documentation (IDRVN) gives some recommendations to name the most usual meteorological quantities used in the models.

6 Presentation norms for namelists.

∗ **Norms:** To make easy the namelists comparison and the namelists update (for a new cycle or a new operational configuration), it is important that the namelists match with a minimal amount of norms. Some of these norms also ensure a good portability of namelists on different type of computers. The norms of namelists presentation can be summarized as follows:

- rule 1/ All the namelists elements should be in the alphabetical order (this is the order obtained when applying the procedure “xpnam”).
- rule 2/ All the existing elements should be referenced (that includes all NEM... elements which are read only in ALADIN), even if they are not read in the current configuration.
- rule 3/ Each referenced element appears only once.
- rule 4/ No obsolete element should appear.
- rule 5/ For logical variables, one always writes .TRUE. or .FALSE.; for example one should write “LREGETA=.FALSE.,”, but never “LREGETA=.false.,”, “LREGETA=.f.,”, “LREGETA=.F.,”, “LREGETA=F,”.
- rule 6/ Only uppercase letters are allowed; for example one should write “NRADFR=-1,”, but never “nradfr=-1,”.
- rule 7/ Each line should end by a comma.
- rule 8/ No blank character is allowed before or after the sign “=”, and before the final comma; for example one should write “NRADFR=-1,”, but never “NRADFR =-1,”, “NRADFR= -1,” or “NRADFR=-1 ,”.
- rule 9/ In each element, there is one (and no more) variable per line.
- rule 10/ Each referenced variable appears only once.
- rule 11/ No obsolete variable should appear.
- rule 12/ Blank characters are allowed before the name of the variable, but all variables should be aligned. The current standard is one blank character before &NAM... and “slash”, three blank characters before each variable. Tabulations are prohibited.
- rule 13/ For arrays, when elements are filled one by one, (for example NFPRGRI), elements appear with order 1, 2, 3, 4 (increasing order): for example finding NFPRGRI(2) between NFPRGRI(29) and NFPRGRI(30) is not allowed; finding another variable between NFPRGRI(29) and NFPRGRI(30) is not allowed.
- rule 14/ For arrays, the following syntaxes are allowed:

```
- syntax 1
 NARRAY=1,2,3,9,
- syntax 2
 NARRAY(1:4)=1,2,3,9,
- syntax 3
 NARRAY(1:2)=1,2,
 NARRAY(3:4)=3,9,
- syntax 4
 NARRAY(1)=1,
 NARRAY(2)=2,
 NARRAY(3)=3,
 NARRAY(4)=9,
```

The following syntax is not allowed:

```
NARRAY=1,2,
       3,9,
```

- rule 15/ Value given to a namelist variable must match the variable type. For example, TSTEP is a real value, so one must write TSTEP=600., TSTEP=600.0 is accepted too.

There are a couple of specific namelists reading namelists of projects other than ARPIFS, for specific applications, which have not to match the rule 2.

∗ **Normalisation tools:** Some tools are available on different machines (at least at METEO-FRANCE) to make the namelists compliant with the previous norms.

- The procedure "xpnam" (available on BEAUFIX) reorders the namelist elements in the alphabetical order, and removes the duplicate elements.
- Empty namelists containing only the right elements can be found on MEROU, for example: /home/mrpm603/ykscar/name_cy42_vide
- The procedure "alignnamelist" (available on BEAUFIX, path /cnrm/gp/mrpm/mrpm603/ykproc/alignnamelist) allows, when providing an empty reference namelist with the right elements, and a namelist to be corrected:
  - to reorder the namelist elements in the alphabetical order.
  - to remove the obsolete elements.
  - to add the missing elements.

7 Dependencies and hierarchy between each project: current status, what is allowed, what is forbidden.

∗ **Groups of projects involved to make executables:** We should be able to make executables with only the following subsets of projects:

- ARPIFS + TRANS + IFSAUX + ALGOR + MPA + MSE + SURFEX: for ARPEGE forecasts with METEO-FRANCE physics.
- ARPIFS + ALADIN + TRANS + ETRANS + IFSAUX + ALGOR + BIPER + COUPLING + MPA + MSE + SURFEX: for ALADIN or AROME forecasts.
- ARPIFS + TRANS + IFSAUX + ALGOR + SURF: for IFS forecasts with ECMWF physics.
- ARPIFS + TRANS + IFSAUX + ALGOR + MPA + MSE + SURFEX + BLACKLIST + ODB + SATRAD + AEOLUS: for ARPEGE assimilations with METEO-FRANCE physics.
- ARPIFS + ALADIN + TRANS + ETRANS + IFSAUX + ALGOR + BIPER + COUPLING + MPA + MSE + SURFEX + BLACKLIST + ODB + SATRAD + AEOLUS: for ALADIN or AROME assimilations.
- ARPIFS + TRANS + IFSAUX + ALGOR + SURF + BLACKLIST + ODB + SATRAD + OBSTAT + AEOLUS + SCRIPTS: for IFS assimilations with ECMWF physics.

This possibility of using only a subset of the projects makes some restrictions about the interdependencies between the projects.

∗ **Hierarchy between the projects:** The rules which should be observed in theory, in order to keep the possibility to easily make the executables, to avoid unmanageable dependencies, and to keep most of the projects as black boxes (as self-containing as possible), can be summarized as follows.

May project A call routines of project B?

- y: yes, in all cases.
- yr: yes, with some restrictions.
- yo: yes, but in a task doing assimilation, screening or trajectory update (and more generally observation treatment) only.
- yl: yes, but in a LAM task only.
- n: no.
- s: sometimes, in specific cases.
- cnr: may be possible in theory (with some restrictions), but currently not relevant.

project A	ARPIFS	ALADIN	SURFEX	MPA	MSE	SURFEX	AEOLUS	BLACKLIST	OBSTAT	ODB	SATRAD	SCAT	TRANS	ETRANS	BIPER	COUPLING	UTILITIES	ALGOR	IFSAUX
project B																			
ARPIFS	y	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
ALADIN	yl	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
SURF	y	cnr	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
MPA	y	y	n	y	s	n	n	n	n	n	n	n	n	n	n	n	n	n	n
MSE	y	y	n	s	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n
SURFEX	n	n	n	n	y	y	n	n	n	n	n	n	n	n	n	n	n	n	n
AEOLUS	yo	yo	n	n	n	n	y	n	n			n	n	n	n	n	n	n	n
BLACKLIST	yo	yo	n	n	n	n		y	n			n	n	n	n	n	n	n	n
OBSTAT	yo	yo	n	n	n	n		n	y			n	n	n	n	n	n	n	n
ODB	yo	yo	n	n	n	n		n	n	y		n	n	n	n	n	n	n	n
SATRAD	yo	yo	n	n	n	n		n	n		y	n	n	n	n	n	n	n	n
SCAT	yo	yo	n	n	n	n		n	n			y	n	n	n	n	n	n	n
TRANS	yr	n	cnr	cnr	cnr	n	cnr	cnr	cnr	cnr	cnr	cnr	y	n	n	n	yr	n	n
ETRANS	n	yr	cnr	cnr	cnr	n	cnr	cnr	cnr	cnr	cnr	cnr	n	y	n	n	n	n	n
BIPER	yl	y	cnr	cnr	cnr	n	cnr	cnr	cnr	cnr	cnr	cnr	n	n	y	n	n	n	n
COUPLING	yl	y	cnr	cnr	cnr	n	cnr	cnr	cnr	cnr	cnr	cnr	n	n	n	y	n	n	n
UTILITIES	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n
ALGOR	y	y	y	y	y	n	y	y	y	y	y	y	y	y	y	y	y	y	n?
IFSAUX	y	y	y	y	y	n	y	y	y	y	y	y	s	s	y	y	y	s	y

May project A use modules of project B?

- y: yes, in all cases.
- yr: yes, with some restrictions.
- n: no.
- nr: not relevant (no module in project B).

project A	ARPIFS	ALADIN	SURF	MPA	MSE	SURFEX	AEOLUS	BLACKLIST	OBSTAT	ODB	SATRAD	SCAT	TRANS	ETRANS	BIPER	COUPLING	UTILITIES	ALGOR	IFSAUX
project B																			
ARPIFS	y	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
ALADIN	nr	nr	nr	nr	nr	n	nr	nr	nr	nr	nr	nr	nr	nr	nr	nr	nr	nr	nr
SURF	n	n	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
MPA	n	n	n	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
MSE	n	n	n	n	y	n	n	n	n	n	n	n	n	n	n	n	n	n	n
SURFEX	n	n	n	n	y	y	n	n	n	n	n	n	n	n	n	n	n	n	n
AEOLUS	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n	n	n	n	n
BLACKLIST	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n	n	n	n
OBSTAT	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n	n	n
ODB	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n	n
SATRAD	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n	n
SCAT	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n	n
TRANS	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n	n
ETRANS	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n	n
BIPER	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n	n
COUPLING	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n	n
UTILITIES	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n	n
ALGOR	yr	yr	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	y	n
IFSAUX	y	y	y	y	y	n	y	y	y	y	y	y	y	y	y	y	y	y	y

Additional remarks about the hierarchy:

- Routines of project ARPIFS must not directly call a routine of ETRANS (an ALADIN routine may be required): indeed this stringent point needs to be discussed again.
- Routines of project ARPIFS may call routines of all other projects. When a project has a specific directory "external", only routines in "external" may be called from a routine of ARPIFS.
- For projects having a specific directory "external", only the routines of "external" may be called from a routine of another project.

∗ **Particular rules for namelists, functions, declaration modules and type definition modules:** For each project:

- namelists are generally in a directory named "namelist".
- only a subset of projects have namelists.
- functions are generally in a directory named "function", or sometimes "include".
- modules are generally in a directory named "module".

OOPS rewritings may lead to modify these rules in the future.

∗ **Current status of the code:** The code is currently not always compliant with all these rules. The consequence is that we sometimes encounter some misleading dependencies when compiling code and doing executables. Another consequence is the difficulty to search all the callers of a routine or a module which seems useless: there is a danger to remove a deck which seems unused, but which is actually still used in a location not easy to find. TRANS, ETRANS and SURF provide examples of well written externalized applications; this is not the case of projects like ODB which is currently not well externalised (calls ARP/IFS routines and uses arp/module modules).

∗ **How to do a good project externalization:** We should recall some rules when externalizing some routines and creating a new project: this is not only moving routines from an existing project to a new one. Some modifications in the data flow must be done in order to avoid forbidden dependencies. In particular, the following rules should be matched (these rules are a summary of what I observed in projects TRANS, ETRANS and SURF.

- The new project has its own set of declaration modules (and type definition modules): this new project uses only variables of its own sets of modules. It is allowed to use some IFSAUX modules like PARKIND1 and YOMHOOK. It is desirable for this project to have its own set of prefixes for the declaration modules (avoid YOM, YOE, YEM, YOS_ which are already used). These modules are in a directory "module".
- Mixed modules (containing type definition, variables declarations and encapsulated code) should be in a deck, the name of which ends by "_mod.F90".
- The new project has if possible a directory called "external" where all the routines which can be called from another project are stored. The other directories contain routines which cannot be called from another project.
- Calling ARPIFS or ALADIN routines from this project should be avoided if possible.
- Some other forbidden dependencies may occur according to the purpose of the new project.
- The new project can have its own set of namelists: this new project only reads its own set of namelists, and these namelists are not read by any other project. It is desirable for this project to have its own set of prefixes for the namelists (avoid NAM, NAE, NEM and more generally all the three letter prefixes already used in other projects). The first letter should be "N".
- The new project should be compliant with the DOCTOR norm. When non-DOCTOR imports are done, it is desirable to clean these imports in order to make them DOCTOR-compliant. The same rules apply concerning the free format F90 compliance.
- It is desirable to write a tester; externalisation can be achieved only if one is able to write an easy tester.
- Projects TRANS, ETRANS and SURF provide examples of a clean way to implement new projects.

8 References and documentation.

- (IDBAS) Yessad, K., 2015: Basics about ARPEGE/IFS, ALADIN and AROME in the cycle 42 of ARPEGE/IFS (internal note).
- (IDCRT) El Khatib, R., 2003: Coding standards for ARPEGE/IFS/ALADIN. Internal note, 42pp, available on "http://www.cnrm.meteo.fr/gmapdoc/".
- (IDCS11) Rivière, O., C. Fischer and M. Fisher, 2011: Coding standards for ARPEGE, IFS and ALADIN systems (V6). Internal note, 10 pp, available on "http://www.cnrm.meteo.fr/gmapdoc/".
- (IDODBO) Kertész, S., 2001: A short overview of ODB. Internal note, 14 pp, available on "http://www.cnrm.meteo.fr/gmapdoc/".
- (IDODBT) Puech, D., 2009: Documentation technique sur ODB. compilation of internal notes (generally in French) available on "http://www.cnrm.meteo.fr/gmapdoc/".
- (IDODBU) Kertész, S., 2002: Using ODB in ALADIN. Internal note, 21pp, available on "http://www.cnrm.meteo.fr/gmapdoc/".
- (IDODBZ) Fouilloux, A., 2013: ODB documentation (several .tar files, restricted access on ECMWF intranet).
- (IDRVN) Yessad, K., 2015: Recommended variable naming in ARPEGE/IFS. Internal note.
- (IGIT) Martinez, S., and C. Fischer, 2013: Météo-France's GIT toolbox: GIT tools (version 1). Internal note, 22pp, available on "http://www.cnrm.meteo.fr/gmapdoc/".