RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE

GMKPACK, 16 years of a build system

Ryad El Khatib
1st ACCORD workshop in visio-conference, 12-16 April 2021

Plan

1. Genesis
2. Past evolutions & milestones
3. Strengthes and Weaknesses
4. Last innovation : the « hub »
5. Possible future developments
6. Conclusion

1. Genesis : early years 2000

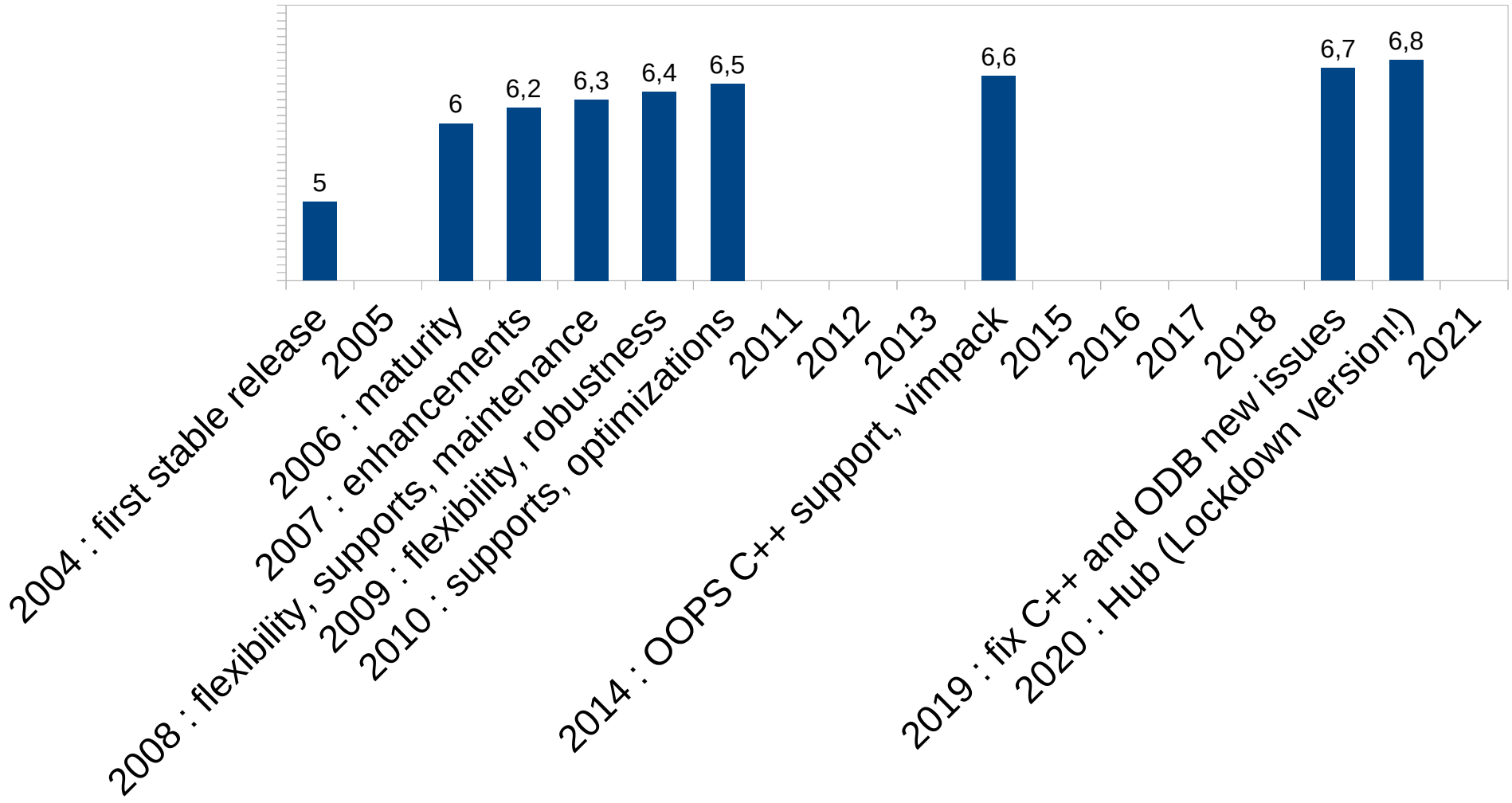
- The increasing complexity of the source code, with several langages and the emergence of F90 dependencies made the initial build system « cf90 *.F » insufficient

- « mkpack », based on ‘make’ by the team GCO and suitable for administrators
  - Had to be wrapped for daily developpers
  - Eventually failed to handle certain duplicated files between aladin/ and arpege/

- « gmak » written in perl and from scratch by the new team GCO
  - Solved the issue with duplicated files
  - Was not user-friendly for daily developers

- => GMKPACK : a wrapper of gmak for developpers

2. Past evolutions & milestones



3. Strengthes & Weaknesses

- Portable, limited maintenance centralized in Météo-France for many partners
- Acclaimed by developers (as far as I can see & read)
- Though it is a « special » build system, vendors are OK with it
- Backward compatibility with older source code versions
- Sufficiently fast
- Self-contained « source code manager of the poor »
- Easy navigation in the code

- Source code and object codes are mixed (no pure « build » directory) making the pre-processing operations complex
- Dependencies research and update not (yet) optimal
- Construction of include path a bit fragile
- Surely not enough documented
- A gaswork

4.1 Last innovation : the « hub »

- External libraries are released with their own build process (make, cmake, etc) and **we don’t want to re-invent the wheel** if that build system is robust *(this is not always the case, unfortunately)*
- The way these libraries are compiled should fit the way our code is compiled. This is particularly true when C++ code is involved : **certain critical compiler options have to match.**
- Each library comes with its own coding and building rules, which can differ from the way how our code is built **(this is the real reason why a build system is so difficult to write)**
- => The « hub » : a plug-in to host external libraries built with their own build system inside gmkpack in order to keep full control of compiler options consistency

4.2 How to make a « hub » … or not

gmkpack …	gmkpack -K …	gmkpack -K -k …	gmkpack -k …
sys/	sys/		
src/	src/	hub/	
lib/	lib/	ics_packages	
bin/	bin/		
ics_masterodb	ics_masterodb		
	hub/		
	ics_packages		
Business as usual	*With hub*	*Hub only*	*Empty !!*

4.3 Example : eckit/fckit and atlas

```
hub/
    local/
        src/
            ecSDK/                    Project #1
            Atlas/
        build/                        Project #2
            ecbuild/
            eckit/
            fckit/
            atlas/
        install/
            ecSDK/
            Atlas/
```

4.4 Example : configuration of fckit

```
GMK_CMAKE_fckit =

# Disable any cmake preset flags :
-DCMAKE_BUILD_TYPE=NONE

# Use gmkpack compiler and flags to drive cmake :
-DCMAKE_CXX_COMPILER=\${CXXNAME}
-DCMAKE_CXX_FLAGS=\"${VCCFLAGS} ${OPT_VCCFLAGS} ${MACROS_CXX}\"
-DCMAKE_Fortran_COMPILER=\${FRTNAME}
-DCMAKE_Fortran_FLAGS=\"${FRTFLAGS} ${OPT_FRTFLAGS}\"

# Fckit needs ecbuild and eckit from the hub
-DCMAKE_PREFIX_PATH=\${TARGET_PACK}/\${GMK_HUB_DIR}/\$
{GMK_LAST_HUB_BRANCH}/\${GMK_HUB_INSTALL}/ecSDK

# ECMWF additional software options (ON/OFF) :
-DENABLE_FINAL=OFF

# Link executable with fckit library taken from the hub :
LD_USR_FCKIT = fckit
```

4.5 Before sailing away with the hub

- Read «HOW_TO_USE_THE_HUB » in gmkpack
- For now, only *cmake* is supported
  but other tools (*configure & make*, etc) can be implemented
- The hub management can hardly be as flexible as for the ‘traditional source code’ :
  - Versions but no « incremental branches » for recompilation
  - No dependencies analysis between the source codes in the hub and the « traditional » source code

5. Possible future developments

- After auto-generated interfaces and python-preprocessed .fypp files, more preprocessing of files (like for Single Column Abstraction ...) could be handled if needed.
- « Universal packs » containing single and double precision executables … but is it worth doing such a development ?
- Extend the usage of the hub to more libraries (those which will be independent enough from Arpege/IFS/Arome)

Whatever new developments will be required,
they have to be anticipated
because time for development *and testing* is needed
while speed and user-friendliness should not be compromised

6. Conclusion

- Gmkpack still good for operations as well as R&D
  16 years after its creation
  See you in 4 years to celebrate its 20th birthday 
- Regular and limited maintenance, but major developments needed every 4-5 years due to source code evolutions
- The new « hub » facility should simplify the link with external softwares
- There is room for new developements, but anticipation is necessary to develop ***and test*** the new features

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

METEO FRANCE

Thank you for your attention !