SAS medium and long term strategy

Carlos GABRIEL

XMM-Newton Science Operations Centre – ESAC / ESA

Current SAS maintenance model until now

- Small team dealing not only with most of the SAS maintenance, but also with all of the PPS
  - only possible at the high level due to experience and excellence of team
- Distributing SAS in many binaries (32- + 64-bit versions, many Linux & Mac versions)
  - making easy its installation to the final user
  - maximising scientific return
  - … but also increasing workload on our side

>> need to redirect efforts… first steps:

- reduce number of platforms
- simplify SAS building procedures
- start thinking in the long-term (*aka post-operational phase*)

SAS 16 (released Jan 2017): breakthrough

- few binaries, no 32bit versions anymore
- no NAG Fortran compiler dependency anymore

SAS 16.0 released on 16/1/17 - Main changes

GENERAL

- SAS compiles with GNU GCC 6.2, including **gfortran 6.2** (away from NAG fortran compiler)
  Compliance with newest C++ and Fortran coding standards

--- Main element of this release, it implied a very large effort by SAS team ---
(a pre-requisite for future SAS compilation at user's side...)

EPIC

- new task evqpb creates for any given EPIC (FF) science exposure an event list containing the associated quiescent particle background (QPB) - (using FWC data) >> correction of spectra and/or images
- new metatask edetect_stack for EPIC source detection on overlapping fields of different observations
- new task implotregions > EPIC thumbnail images showing selected src and bck extraction regions
- ebkgreg determining most suitable position of background selection region for a source > now for MOS
- eslewchain is now including source detection in Slew Data
- the task collection esas for extended source analysis, now fully coded in F90

RGS

- upgraded task rgsrmfgen, including new (optional) RGS effective area correction
- new task rgsimageforbadpixfind ancillary to implementation of alternative RGS bad column finding

+ **many fixes and improvements:** 90 tasks upgraded, with 229 sub-version changes

SAS 16.0 released on 16/1/17 - gfortran migration

List of S/W changes needed for migration:

- ★ new definition of array descriptors (dope vectors)
- ★ memory mapping of array descriptors
- ★ use of specific MACRO statements for NAG
- ★ different naming convention for precompiled modules
- ★ usage of intrinsic functions - cases where conversion depends on the compiler
- ★ get_environment variable, leading blank spaces removed properly
- ★ implied_loop - different standards regarding manipulation of arrays in loops
- ★ use of reserved words - gfortran more strict
- ★ integer to string conversion - gfortran more strict
- ★ namespace errors - gfortran more strict regarding the scope of module names
- ★ memory allocation - optional parameter as part of allocatable variable def not possible in gfortran
- ★ array initialisation - non initialised arrays in gfortran are undefined
- ★ runtime errors - gfortran is more strict with definition of parameters in subroutines
- ★ overflow computations - gfortran more restrictive regarding size of variables
- ★ pointers - in gfortran all pointers passed to C++ must be initialised to null
- ★ different treatments of NaNs

SAS 16 release - binaries

From SAS 16.0 on **no** 32bits binaries anymore
+ reduction to few 64bits binaries

	OS	Kernel	libc
Linux 64:	**RHEL 6.8**	**2.6.32**	**2.12**
	Ubuntu 16.04.1LTS	**4.4.0**	**2.23**

MacOS:	**MacOS 10.10.5 (Yosemite)**	**Darwin 14.5.0**	**1213.0.0**
	MacOS 10.11.6 (El Capitan)	**Darwin 15.6.0**	**1226.10.1**
	MacOS 10.12.3 (Sierra)	**Darwin 16.3.0**	**1238.0.0**

Alreadv in SAS 15

New in SAS 16

+ 1 universal SAS-VMs (64bits) - Ubuntu16.04.1

Medium- and long-term strategy

On top of "normal" maintenance (and development)... working at the same time on Post-ops

- to be better prepared if something leads to termination of XMM-Newton
- to reduce the work which will be needed for legacy during the post-operational phase (limited strictly to 2 years)

Main ideas wrt SAS data processing after EoM:

A. preserve a running SAS as long as possible
   - A1) SAS Virtual Machine … Dockers *(≥ 10 years)*
   - A2) RISA (Remote Interface for Science Analysis) *(5-10 years longer)*

B. give SAS code to community *(re-use? … further development?)*

To make possible B means: reducing complexity = modernising

- B1) Compilers: maintaining close correspondence with new compilers
- B2) Migration to Python in 3 areas: graphical, replacing PERL, replacing calls to HEASOFT
- B3) Simplify configuration and improve documentation: making possible / easy building from source & source maintenance

Three years detailed plan (2017-2020) based on these lines

SAS VM + RISA == SAS on the net

A1 - SAS is distributed since 2006 also as Virtual Machine

Estimation: such a VM could run after EoM in the most diverse OS's for ≥ 10 years

Proven: SAS-VM (2006) running today without any problem on actual OS's

A2 - RISA is a fundamental component in our long-term strategy ...

it is already working though:

- on-the-fly reprocessing of archival data with latest SAS and calibration
- filtering and light data reduction services

>> integration in XSAv9.4

>> first steps in the way to a more complete / full RISA I/A service

RISA post-ops thought so far to be SAS-VM based ...
(> final SAS packed in one OS… extended life [5-10 years] in a central place)

XMM Solaris 8 operational machines (from 2004!) replaced these days…

>> RISA is ideal system for experimenting replacement of SAS-VM by Dockers… (2018)
(Dockers would ease combination of SAS data reduction with other S/W)

Compilers, configuration and builds

B1) SAS 16: Transition to GNU provided gfortran compiler

at the same time, most modern C++ compiler version used: GCCv6.2

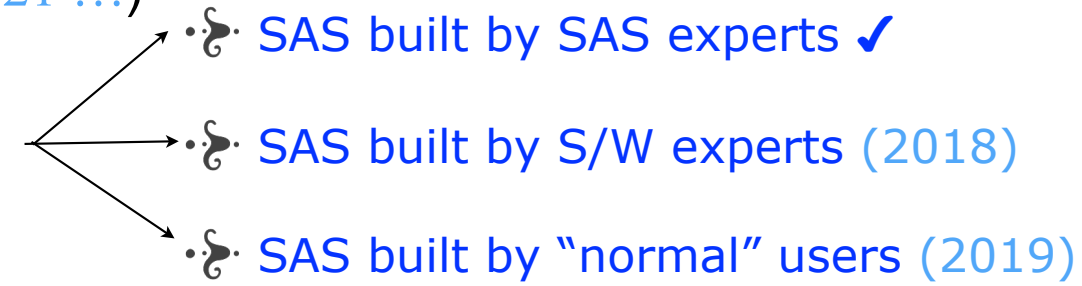
>> serious scrutiny of source code
>> most up-to-date standards both in Fortran 90 and in C++

Pre-requisites for providing source code for SAS compilation at user's side

- free compilers ✓
- code up-to-date with standards ✓ (… 2019 … 2021 …)

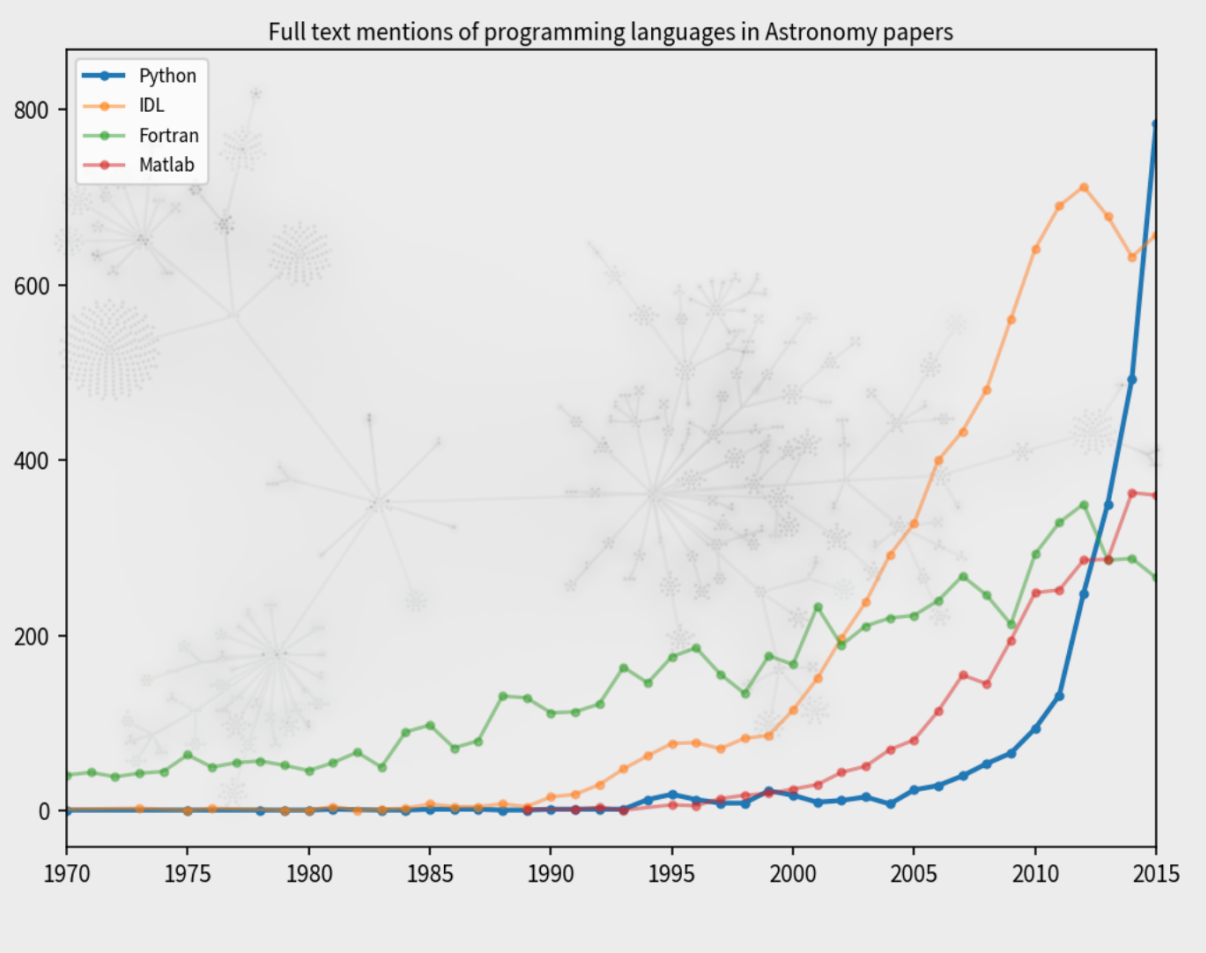
+

- simplified configuration and building procedures
  + documentation



Moving to Python

B2 - Moving certain SAS areas to Python



Robitaille et al 2017 67% of all astronomers use Python

Why Python?

- Simplifying SAS & PPS
  >> more maintainable
- Pre-condition for a future package to be given to the community

Stepwise introduction in SAS/PPS:

t

- graphics area: first products (2017)
- graphics area: replacement of PGPLOT & Grace (2018-19)
- scripting area: replacing PERL (yes, lot of work… 41 scripts in SAS, some pretty complex) (2019)
- Heasoft area: replacing tasks depending on Heasoft (2020)

Making easier to build & maintain a running SAS

B3 - Configuration / build / documentation

Two main problems for "aliens" to deal with SAS on the source code level

- SAS is a complex piece of software >> SAS is difficult to build…
  1. replace NAG fortran compiler by gfortran ✓
  2. simplify build & configuration … extend documentation so that SAS experts can build ✓ (2017)
  3. simplify more … extend documentation so that S/W experts can build (2018)
  4. simplify even more … extend documentation so that 'aliens' can do it (2019)

- To maintain S/W written by others is difficult (don't tell us…!)

  Internal documentation is essential - Improve so that non-experts can cope with

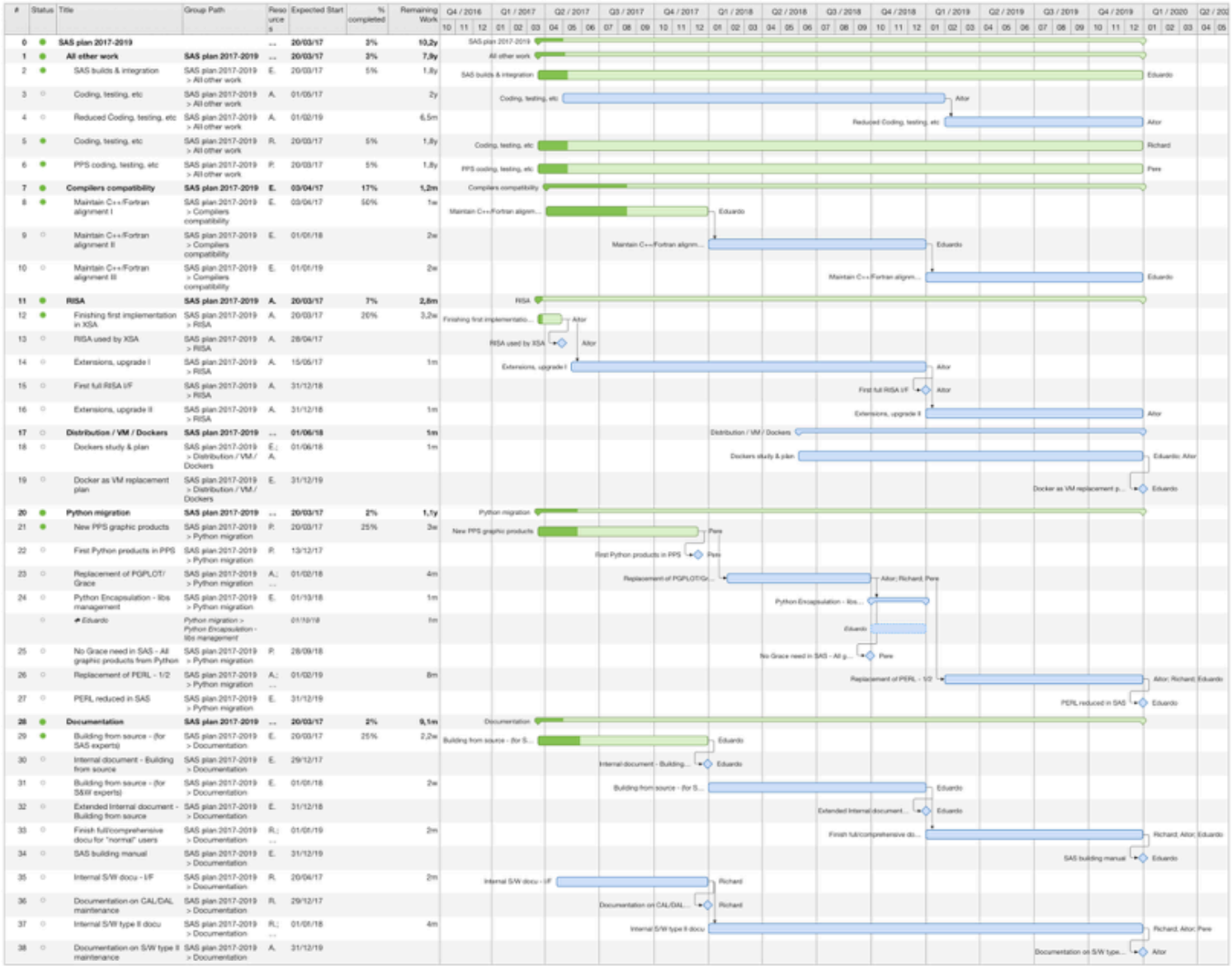
  1. I/F type S/W (OAL, CAL, DAL) (2017-18)
  2. S/W type II (beyond calibrated event lists…) (2019)
  3. Rest of the S/W (2020)

This will be needed even before / independently of Post-ops

The triennium plan: 2017-2019

Main assumptions:

- (almost) all the extra work for Post-Ops can be done by the SAS central team
- only 20% of the time is dedicated to Post-ops
- plan finishes mid 2020, but activities in this framework continue (due to OS evolution, etc)



The triennium plan: team members workload