TELCAL tasks

Jean-Christophe Roche, IRAM(*)

Casa developers meeting
NRAO (Socorro)
10-14 May 2010

(*)Institut de Radioastronomie Millimétrique
Grenoble, France
Plan

- TELCAL subsystem
- Off-line mode
- Task module
- StandAlone module
- Future developments
TELCAL subsystem team

- Robert Lucas (leader)
- Dominique Broguiere
- Jean-Christophe Roche
- Juan Pardo (ATM author)
- Takeshi Nakazato
- Fanny Cosson
TELCAL subsystem overview

- TELCAL = Telescope Calibration
- goal: ensure that ALMA array is and remains in a proper shape
- more than 8 years of development
- ~ 100 000 lines of code
- C++, Java, Python, Fortran languages
- two uses: on-line vs off-line
- on-line: communicate through a CORBA based architecture (ACS)
TELCAL subsystem

calibrations

- **time dependent**
  - amplitude, amplitude curve
  - phase, phase curve

- **frequency dependent**
  - bandpass
  - sideband ratios

- **atmospheric**
  - phase length correction (WVR)
  - temperature scale (based on ATM)
  - skydip

- **antennas / array**
  - antenna position (*)
  - delay
  - focus
  - holography (*)
  - (optical) pointing model (*)
  - pointing
Off-line mode overview

- why:
  - execute calibrations NOT in real-time (on-line mode)
  - (re)run a calibration from data stored on disk
- who:
  - astronomers
  - developers
  - operators
- how:
  - direct Python binding of engines (SWIG): scripts
  - Casa binding (XML / Python): tasks
- what:
  - provide plots of ASDM + ASDM result (matplotlib)
  - see intermediate results (not in the ASDM result)
Off-line mode
module view

processing the calibration:
- execute a calibration on a scan
- store the results

TELCAL/TelCalResults

Engines
C++ / Python

AsdmReader
C++ / Python

Libraries
C++

Tasks
Python

StandAlone
shell / Python

Tools
C++ / Python

interface with the ASDM classes

Casa binding of TELCAL

package generator (without any ACS stuff)

algorithms to process calibrations

specific libraries used by the engines (ATM, TPOINT, ...)

common tools (traces, exceptions, command line parser, ...)

products

C++ / Python

standalone

common
tools
Off-line mode
how it works

Tasks

AsdmReader

read ASDM

create (1)

write ASDM result

disk

process (2)

Engines

use

Libraries

output intermediate results

output traces

...-----------------------------
scanNum = 51; Source
spectraWindowTogetherDelay=1
numChanFit_m=256
Azimuth: -193.1   Elevation: 67.6
substractReferencePhase
substractReferenceDirection
DelayScan::computeDelayAnt()
iant 0 delay -0 +0(seconds)   weight[0]=0
iant 1 delay -2.236e-11 +1.432e-12(seconds)   weight[0]=0
...-----------------------------
Antenna 1
0 0 0 fitRms=2.34933e-12 rad X=-1.979e-03  Y= 2.711e-02 Z=-2.197e-02 m
Results of the leastSquare fit of sum(delay[iscan]-X[iant]*s[iant]/C)**2)
Xalma=-1.9787e-03 +1.031e-04  Yalma=2.71124e-02 +1.941e-04  Zalma=
2.19711e-02 +3.913e-04 meters
Instrumental delay=7.86797e-11 +1.00832e-12
fitRms= 2.349e-12 second
Calibration::~Calibration() : delete 52 scan(s)
Duration 2678400.000s
Ant: DV02 Position: -0.001979 (0.000103)m  0.027112 (0.000194)m -0.021971
(0.000391)m
Ant: DV02 Geocentric: (-0.005460,  0.008222,  0.033531) m
-1.979 (0.103) 27.112 (0.194) -21.971 (0.391) mm
Task module overview

- current context:
  - ALMA-7_1_0-B
  - ACS 8.2
  - Casa 30.0.9860

- developments:
  - C++ / Python wrappers (SWIG)
  - Python scripts (direct binding of engines)
  - Casa binding (XML / Python)
Task module
C++ / Python wrappers

ACS

AsdmReader
Engines

C++ interface

SWIG

Python interface

Tasks

Python scripts

<module>
C++ / Python
ACS Makefile
+ specific stuff

<module>

• _<module>.so
• <module>.py

Python interface

== SWIG making dependencies: ../object/AsdmReader.dswig
== SWIG making C++/Python wrapper: AsdmReader.py AsdmReader.cpp
== SWIG C++ wrapper compiling: AsdmReader.cpp
== SWIG making Library: _AsdmReader.so
Task module
Casa binding

== Making Casa tasks ...
buildtelcaltasks.sh: copying task files to ../lib/python/site-packages/telcaltasks ...
buildtelcaltasks.sh: building the tasks asdmlist asdmplot tc_amplical tc_amplicurve tc_antpos tc_bandpass
tc_delay tc_focus tc_phasecal tc_phasecurve tc_pointing tc_wvr ...
sed: can't read .bak: No such file or directory
buildtelcaltasks.sh: hacking the telcaltasks.py file ...

... Installing Casa tasks into /diska/home/almadev/introot/lib/python/site-packages/telcaltasks ...
buildtelcaltasks.sh: copying task files to /diska/home/almadev/introot/lib/python/site-
packages/telcaltasks ...
buildtelcaltasks.sh: building the tasks asdmlist asdmplot tc_amplical tc_amplicurve tc_antpos tc_bandpass
tc_delay tc_focus tc_phasecal tc_phasecurve tc_pointing tc_wvr ...
sed: can't read .bak: No such file or directory
buildtelcaltasks.sh: hacking the telcaltasks.py file ...

ACS
AsdmReader
Engines
C++ interface
SWIG
Python interface
Tasks
XML / Python buildmytasks
Casa tasks

ACS Makefile + specific stuff
Task module
binding exemple

XML/Python binding exemple
Task module
running a task
Task module

TELCAL tasks

• 2 tasks for the ASDM:
  • `asdmplist`: list the scan(s) and subscan(s) of an ASDM dataset.
  • `asdmplot`: plot the contents of an ASDM dataset.

• 10 tasks for the engines:
  • `tc_<engine>`: process an <engine> calibration on an ASDM and plot the results.
  • `tasks`: `tc_amplical`, `tc_amplicurve`, `tc_antpos`, `tc_bandpass`, `tc_delay`, `tc_focus`, `tc_phasecurve`, `tc_phasecal`, `tc_pointing`, `tc_wvr`
Task module
Problems

- plotting (matplotlib) is really (really) slow under casapy
- special needs for (command line options) buildmytasks
- hardcoded absolute pathes in mytasks.py generated file: can't install / move a directory that contains a mytasks.py file it require to rebuild all the Casa binding
- how to automatically load a set of tasks? 'casapy -c mytasks.py' don't work in 30.0.9860 (it leaves casapy)
- really need to define twice the tasks description (in the <task>.xml and in the task_<task>.py files)?
- many intermediate Casa files (.last, .log files, ...): directory "pollution" so is it possible to use hidden files?
StandAlone module

- goal: define a TELCAL package without any ACS dependencies
- ASDM and TELCAL off-line modules have a standalone build mode
- collect all C++ and Python dependencies in a graph: snakefood & specific tools
- parse and filter the dependency graph
- build duration: ~ 8 min
- package size: ~ 300 MB

== Generating C++/Python graph dependencies ...
  found 3891 nodes in the graph
== Generating C++ dependencies ...
  found 140 C++ dependencies
== Generating Python dependencies ...
  found 142 Python dependencies
== Collecting dependencies ...
  collected 282 dependencies
== Collecting task files ...
  collected 24 task files
== Adding notice and version ...
== Creating the standalone package ...
  created archive ../lib/telcalsa.tar.bz2 (52308655 bytes)
== Removing temporary files ...
Future developments

- **Tasks:**
  - develop new tasks (primary beam, ...)
  - improve Casa integration: remarks, ... are welcome
  - improve plot in casapy (how?)
  - manage outputed traces (C++, Python)
  - homogenize some stuff (task parameter parsing, ...)

- **StandAlone:**
  - manage redundant libraries (ASDM, ATM, ...)
  - provide a package installer (that detect missing (system) libraries, ...)?