Thomas Jürges <thomas.juerges@gmail.com>

Vertex Antennentechnik GmbH
tjuerges@vertexant.de

ALMA / NRAO (Socorro, NM) 2005 - 2011

ALMA Control sub-system

Hardware, design, software, odd ends

- and the ugly things nobody dares to ask about

ALMA hardware – The Stuff

- ALMA Operations Site (AOS) @ 5000m elevation, Operations support facility (OSF) @ 2950m
- Antennas:
  - 50 * 12m (25 by Vertex, 25 by AEM)
  - 12 * 7m, 4 * 12m (by Melco, ALMA Compact Array – ACA)
- Pads at various locations, array configuration changes with every cycle, max. baseline approx. 20km
- Communication / data transmission via CAN-bus, TCP/IP, optical fibre

Location is everything!

Driest desert on earth: the Atacama desert

Typically 1.0mm perceptible water vapour – can be as low as 0.5mm 25% of the time

External software

- Linux Kernel 2.6.29.4, RTAI 3.7.1
  - realtime system for all ALMA computers that execute realtime software
- GCC 4.1
  - compilation of C/C++ code
- Python 2.6
  - Control Command Language
- JAVA 1.5
  - GUIs

External software, contd.

- CALC: FORTRAN package - yuck! -, maintained by staff at JPL.
  - In Control used for calculation of geometric delays
  - Standard in this area
  - Widely used by VLBI community
  - Freely available
- CASA: Official data reduction toolkit for ALMA
  - Freely available (NRAO Socorro, New Mexico)
- SLALIB: Positional Astronomy Library by Pat Wallace
  - FORTRAN version freely available
  - C version not so freely available, has to be licensed
  - The standard – Period

ALMA software development process?

- High distribution of groups all over the world (NRAO, ESO, NAOJ, JAO)
- Once a year leads meeting
  - Planning, design reviews, etc.
  - Preferably in a nice place ;-)
- Good communication is key to everything!
  - Weekly telecons very important
  - Wikis very crucial
    - But not for everybody
- Guidelines for everything – Not so nice!
  - Software guidelines – Nice!
    - Would be even nicer if people followed them. Sheriff necessary!

ALMA software development – only the interesting stuff

- Hardware – Software: Interface Control Documents (ICDs)
  - Sometimes ICD stands for “Ignored Computing Document”:
    - Hardware developers like to modify hardware without updating ICD or telling software developers
    - Firmware gets updated which invalidates some monitor or control points in the ICD
    - --> Great fun for everybody!

The Control sub-system does what?

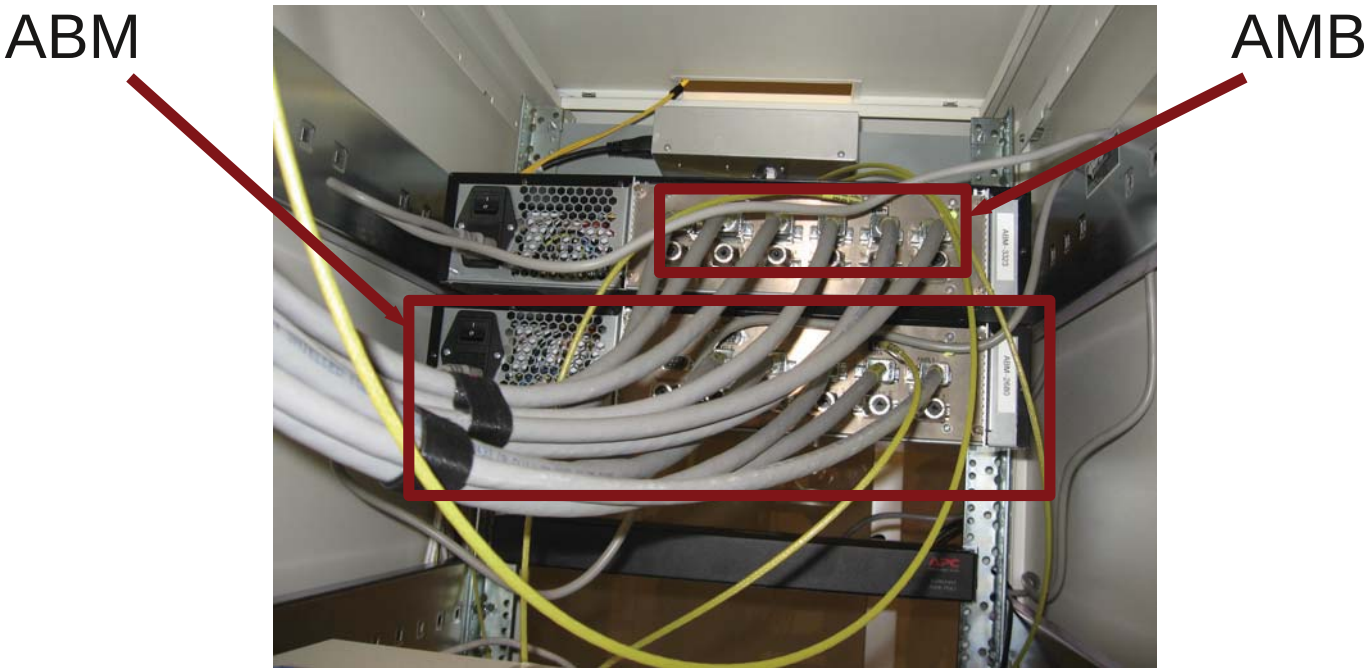
- Responsible for overall control of ALMA telescope
- Coordinates hardware, initiates observations, monitors their progress
  - Most of Control sub-system software runs on ABM hardware

Hardware for controlling the hardware

- The ABM in the Control sub-system

  Antenna Bus Master (ABM), Monitor & Control computer in each antenna
  - VMIVME-7807 (Pentium-M 1.1 – 1.8GHz, 1.5GB RAM), XVB-601 (dual core 2.8GHz, 8GB RAM), both VME
  - ELMA 19” crate
  - Custom back panel (NRAO)
  - 1 GBit Ethernet port
  - 5 CAN-bus channels @ 1MBit (plus one spare / debug which is sometimes in use) -> ALMA Monitor and control Bus (AMB)

ABM - AMB



Begin of commercial break

Baumstraße 44 - 50
D-47198 Duisburg / Germany

Phone: +49 (0)2066 2096-0
Fax: +49 (0)2066 2096-11
Email: info@vertexant.de
Web: www.vertexant.de

VERTEX ANTENNENTECHNIK

PRODUCTS

Precision Antenna Systems

Satcom Products

VERTEX ANTENNENTECHNIK

APPLICATIONS / FIELDS OF ACTIVITIES

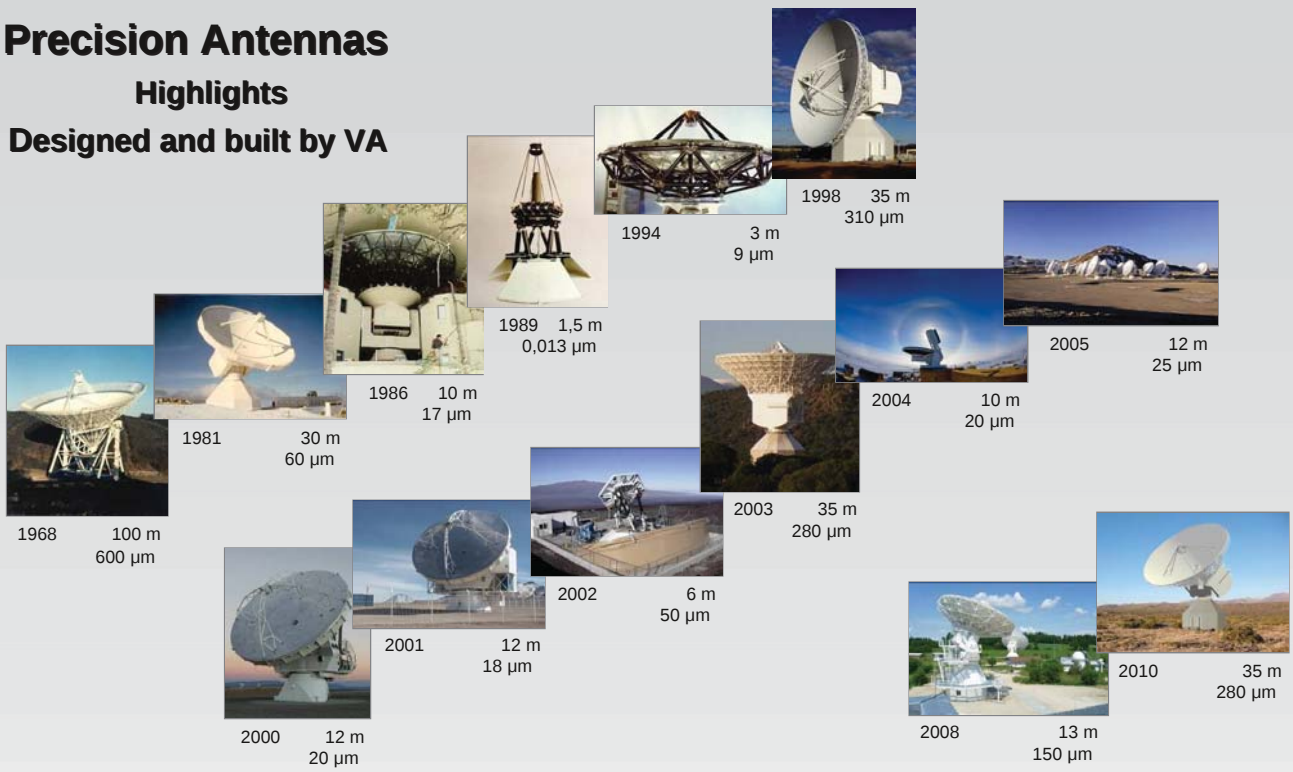
- SATELLITE COMMUNICATIONS
- TT&C / IOT / RANGING
- EARTH OBSERVATION / REMOTE SENSING
- DEEP SPACE MISSIONS
- TRACKING OF FAST MOVING OBJECTS
- TRANSMITTER LOCATION AND IDENTIFICATION
- RADIO AND OPTICAL ASTRONOMY

VERTEX ANTENNENTECHNIK

Precision Antennas

Highlights

Designed and built by VA

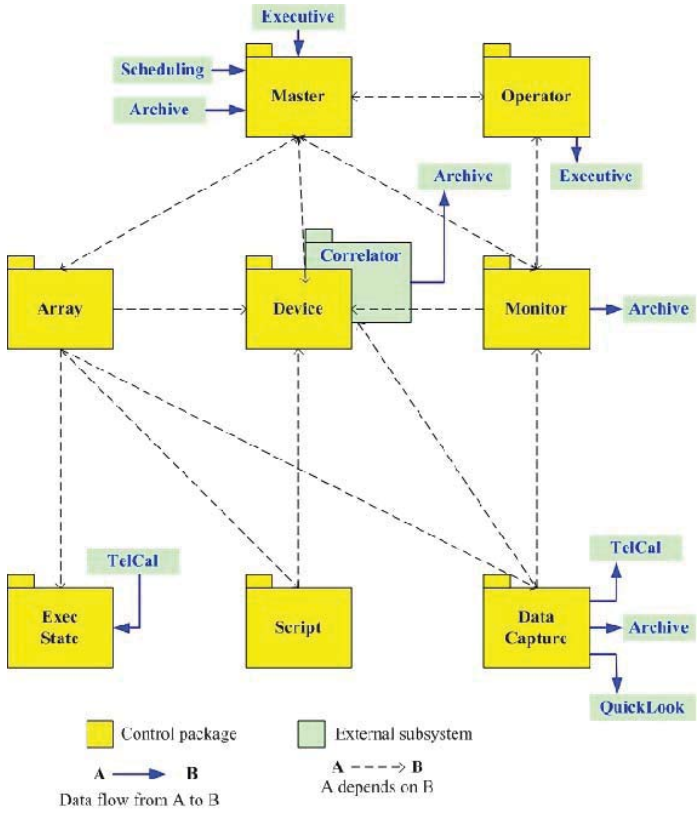


End of commercial break

ACS == ALMA Control sub-system?

- No, but almost; :-)
- Most software developed by ALMA:
  - based on / uses ACS
  - -> So is the Control sub-system.
  - Good! -> Reuse it for other projects!
- Control sub-system uses most ACS features but for BACI properties
  - Not for much longer! Refactoring is underway:
    - Use BACI properties for monitor and control
  - Later? DDS -> Joe :-)

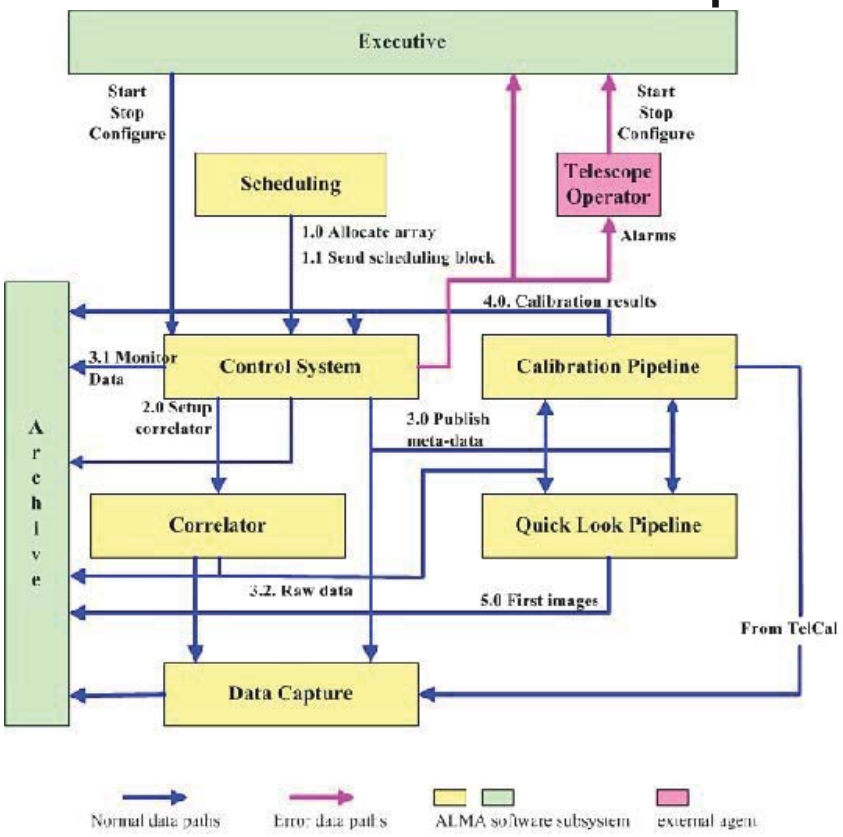
Control consists of stuff...



Control sub-system interacts...

- Archive
- Correlator (Corr)
- Logging
- Executive (Exec)
- Proposal & observing preparation (Obsprep)
- Pipeline
- Scheduler (Sched)
- Telescope calibration (Telcal)

The Control-“U-Bahn” explained



Control controls what again?

- Mostly hardware!
  - How? “Device Drivers” (ACS components)
  - Per-hardware device code is automagically generated.
    - Great!
  - Ìt is generated by a hand coded XML parser written in Java.
    - Um... Not so great – at least for Thomas.
  - The XML files are Excel spreadsheets.
    - Wait a second! What?

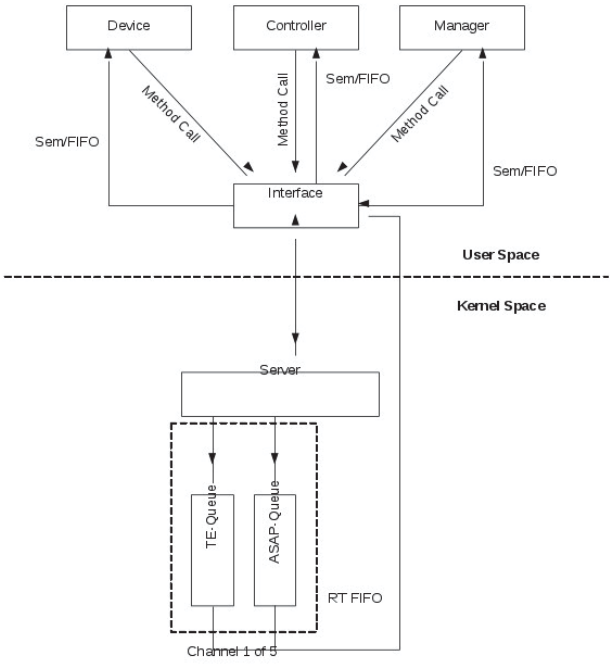
Control Devices – Device Drivers

- Classes for simulation and real hardware inherit from C++ base class
- Implementations (“real beef”) inherit from classes for simulation and real hardware
- Contain (in many cases) only methods and no properties (see below)
- Monitor & control is executed by explicitly calling methods in “device drivers”
- Refactoring of Control devices -> Reuse realtime code from helper libraries.
  - Careful! Pitfalls: global variables, static members, etc. But you knew that already.
- Autogenerated test code tests generated code

Control Devices, contd.

- Higher level “devices” have no hardware representation, but monitor / control groups of hardware devices
- Monitor and Control of the hardware via AMB (ALMA Monitor & control Bus)
- Tests are developed by the authors of the code to be tested
  - -> Problematic: I test only things that I know of to be working.

Realtime?



- Yep, for monitor and control!
- Only on dedicated realtime computers
- Only in the Linux kernel for TE reception and CAN-bus communication

ACS – the big gun for everything?

- Pros:
  - Nice layer on top of CORBA, hiding the ugly everyday CORBA stuff
  - Allows and encourages code reuse (components)
  - Good support when ACS developers have time ;-)
  - Framework that has a lot of useful features already built in
- Cons:
  - Implementation not always complete (BACI property types)
  - CDB sucks – ALMA replaced it with the TMCDB – don't know if it sucks, too, because lack of experience with it.
  - DevIO does not match ALMA use case
  - Makefiles sucked big time – Has gotten much much better!!!
  - Bulk data system before DDS very unreliable

Questions that I started asking

- Is VME useful / really necessary?
- Is CAN-bus something I would have chosen?
- Has C/C++ been a good choice for hardware-related code?
- Is RTAI really the best solution for ALMA?
  - Why not use the vanilla kernel and try it with soft-realtime?
  - If it does not work, use RT-PREEMPT!
  - -> Much cleaner solution
  - What if Paolo gets hit by a bus?
- Why not more code reviews?
- Why not put an emphasis on reliably working code instead of pushing for new features and deliveries?
- Could the software have been developed with the same amount of FTEs without ACS?

Questions I have

- How is your development process laid out?
- Did you agree on a testing framework?
- Have you agreed on how to test your code?
- What about coding standards?
- Do you need realtime?
  - Yes: do you need hard realtime? Where?
- Have you made regular code reviews part of your development process?
- What mistakes have you decided not to make that ALMA made?

The real Q & A