Presentation Abstract

An introduction to concepts, theory and implementation of telescope control systems, with particular attention to antennas for radio astronomy.

I do not know the level of knowledge of the participants to the course: this is more an engineering talk than a science talk.

I assume that the participant are mostly physics and astronomy students.

How many of you are engineers?

How many know control theory?

I will try to steer the presentation based on the questions and my perception of the level of expertise and interest in the different topics.

Please: ask, stop me and tell me if I am going in the wrong direction.

Instructor:

I am Gianluca Chiozzi and I am working at the European Southern Observatory in Munich.

You can reach me at any time at the following email: gchiozzi@eso.org

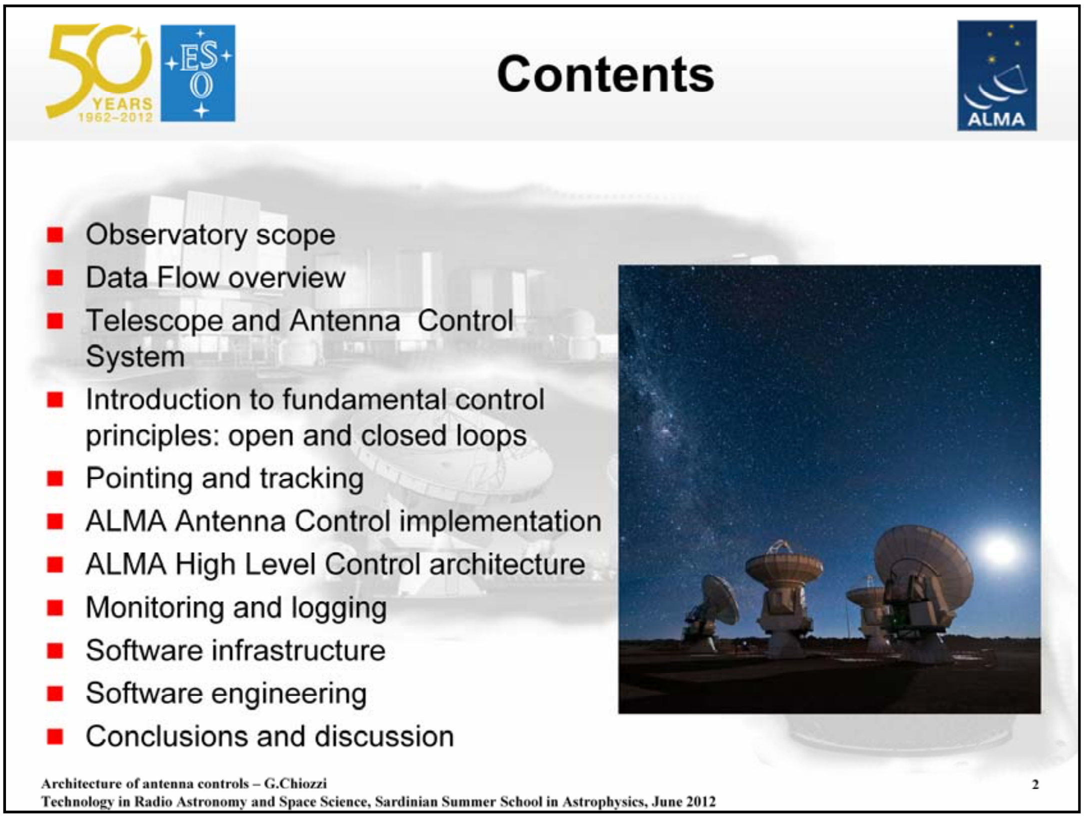
Sine 2007, I am responsible for the ESO Control and Instrumentation SW Department, with about 20 people assigned to different projects (VLT, VLTI, ALMA, E-ELT).

I am also spending some “technical time” in the architecture and design of the E-ELT Control Software and in ALMA SW infrastructure (ACS).

Before that I was responsible for the ALMA Common Software (ACS) architecture and development, with a team of about 10 people (not all full time) distributed in various sites in Europe and North America. ACS is the software infrastructure for the ALMA project and is used also by other projects. I have been working on ACS and in ALMA since about the year 2000.

Before ALMA and ACS I have been heavily involved for about 6 years in the design and implementation of the VLT Common Software and Telescope Control Software. Here I have been responsible for introducing Object Oriented technology in the project, working on the architecture and design of some control subsystems and on the implementation of OO class libraries for the Common Software infrastructure.

Before ESO I have been employed at the IBM Technical and Scientific Research Center in Milan, working on image recognition systems and on user interfaces for utility management systems (like electrical or railways networks).



I will:

- Give a general introduction to the scope of the (antenna) control system in an observatory
- Introduce general theoretical concepts on control systems
- Analyze the specifics of pointing and tracking theory, trying not to repeat what Pablo will say in the following presentation
- Go into a description of the implementation for the ALMA Antenna Control system, with some comparison with other systems.
- Zoom out to the global telescope control system and its interactions with the other observatory subsystems.
- Describe monitoring and logging as essential for a distributed system

The last couple of topics are just outlined in the presentation, because most probably there will be no time.

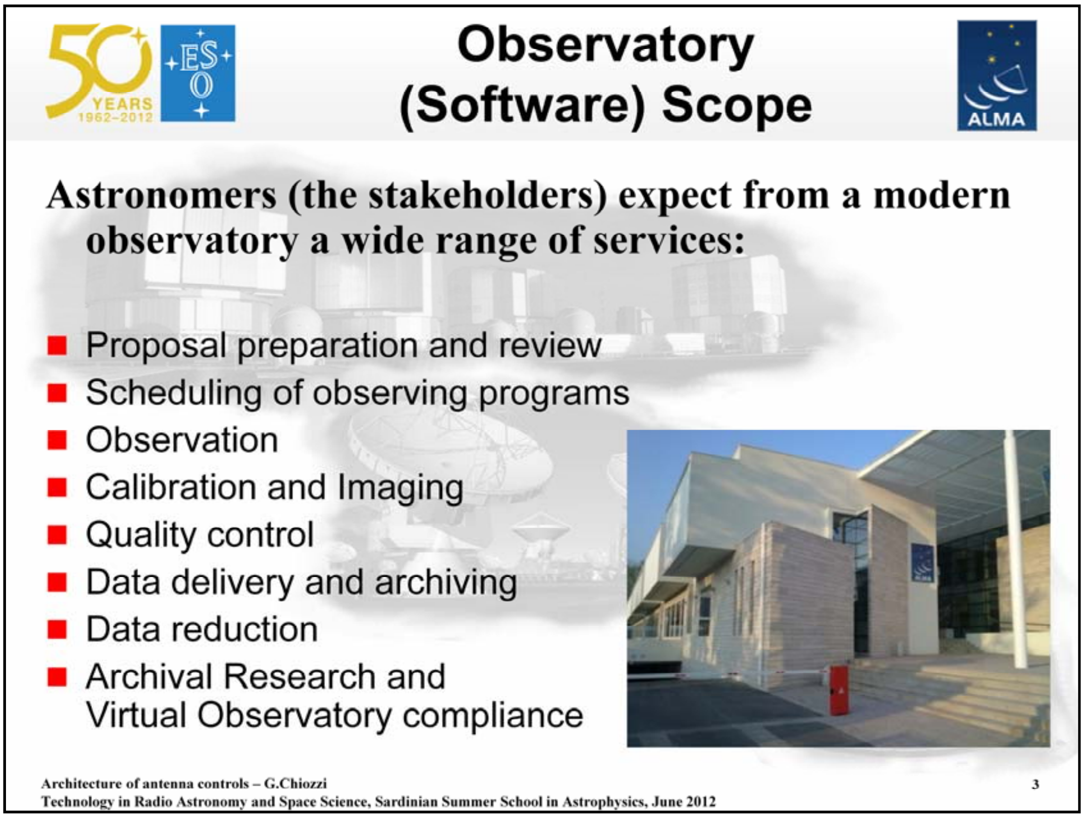
In case, I will talk of them expanding what written in the slides, following them as an outline.

At the end there will be hopefully time for some questions and for a discussion.

I will be in any case very happy to talk with all of you in the next couple of days.

Other topics I could have been talking of, but there will be for sure no time:

- Time distribution
- Portability and maintainability
- Simulation
- Modeling



Astronomers (the main stakeholders for our systems) expect from a modern observatory a wide range of services.

In the past, the Astronomer was traveling to the Observatory, making his observations, storing data (pictures of, afterwards, on tape) and going back home to reduce it.

The telescope and, eventually, the instruments had a control system virtually independent from everything else.

Data reduction was done offline after the observation and there was no direct feedback from the observation data to the control system. An experienced observer was just driving the telescope based on his own feelings.

There was no observation data archive, no quality of service measures and constraints, no facility engineering in the terms we think of now.

This has dramatically changed in the past 20 years, with the big observatories like ALMA, VLT, Keck, Gemini, Hubble and so on.

Since the major observatories are now providing integrated facilities, astronomers expect the same also from smaller ones and the amount of integration required for the new projects like ALMA and the giant optical telescopes like E-ELT, TMT or GMT will be even more. The Virtual Observatory is also contributing to this need for integration and quality control adding the intra-observatory dimension to the problem.

Now all the systems in an Observatory are fully integrated.

Also, astronomical observation is not limited to experts in the field (like infrared or radio astronomers), but it shall be open to chemists, biologists and other multi disciplinary researchers.

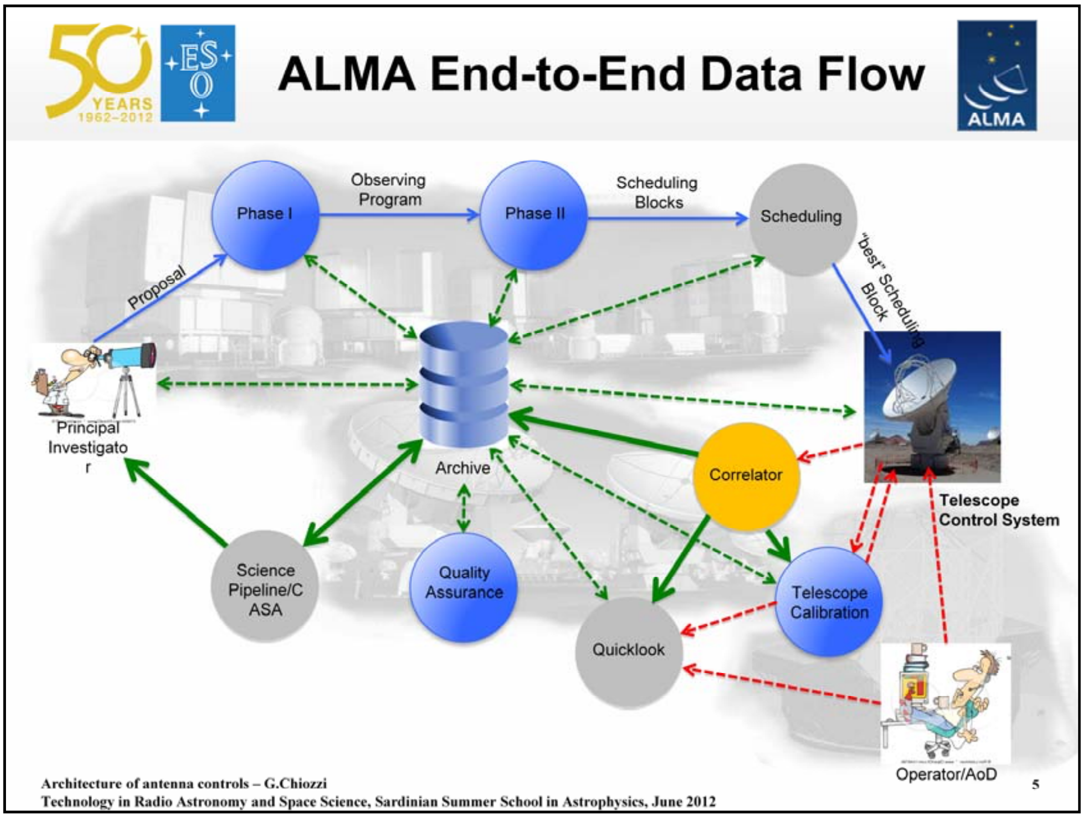
The general user should be given standard observing modes to achieve project goals expressed in terms of science parameters, rather than technical quantities. But experts must be able to exercise at the same time full control.

Making things easy and flexible for the astronomer adds up complexity to the software development

Most of there services are implemented in software or rely heavily on software.

The Modern Observatory

- Data/images are ultimately acquired by the (optical or radio) telescope
- … but there is the need for high integration with the other facilities in an end to end data flow

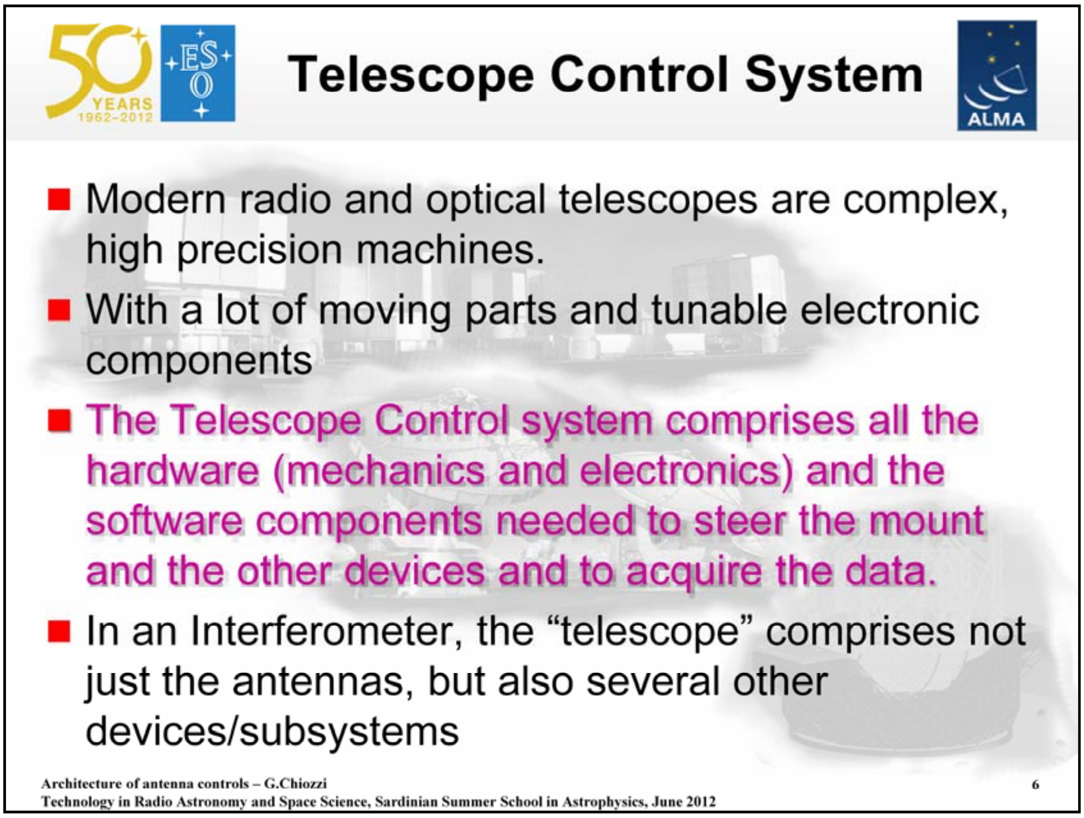


The End-to-End Data Flow of an astronomical observation program using ALMA is a complex process, that includes:

- The process of defining and approving the observing program.
  This identifies the observing requirements, like object visibility, atmospheric conditions, hardware availability, project priority
- The splitting of the program in minimal observation units (the scheduling blocks) that can be executed independently.
- The scheduling process that picks at any time the best scheduling blocks for execution, based on the requirements defined
- The actual execution on the telescope by the control system, using the hardware (antennas and correlator for an interferometer)
  This includes also the selection, execution and control of proper calibrations and allows operator and astronomer
  interventions based on quick look data
- Quality control and quality assurance of the observed data, using standardized and reproducible receipts
- Delivery of the raw data to the observing team and data reduction using standard or specific data processing pipelines.

In ALMA it has been decided to give a very central role to the ALMA Archive where all data needed by the various phases
Is stored and where it can be retrieved by the processes executing the following phases.

In the tight feedback interaction between antenna/array scheduling, control system, correlator, quicklook and pipeline, events
(publisher/subscriber paradigm) are used to synchronize the various processes and activities.



Some ALMA Antenna requirements

• It shall be possible to move with the following speed and acceleration:

Maximum Azimuth angular velocity: > 6 deg/s

Maximum Elevation angular velocity > 3 deg/s

Maximum Azimuth angular acceleration > 18 deg/ s2

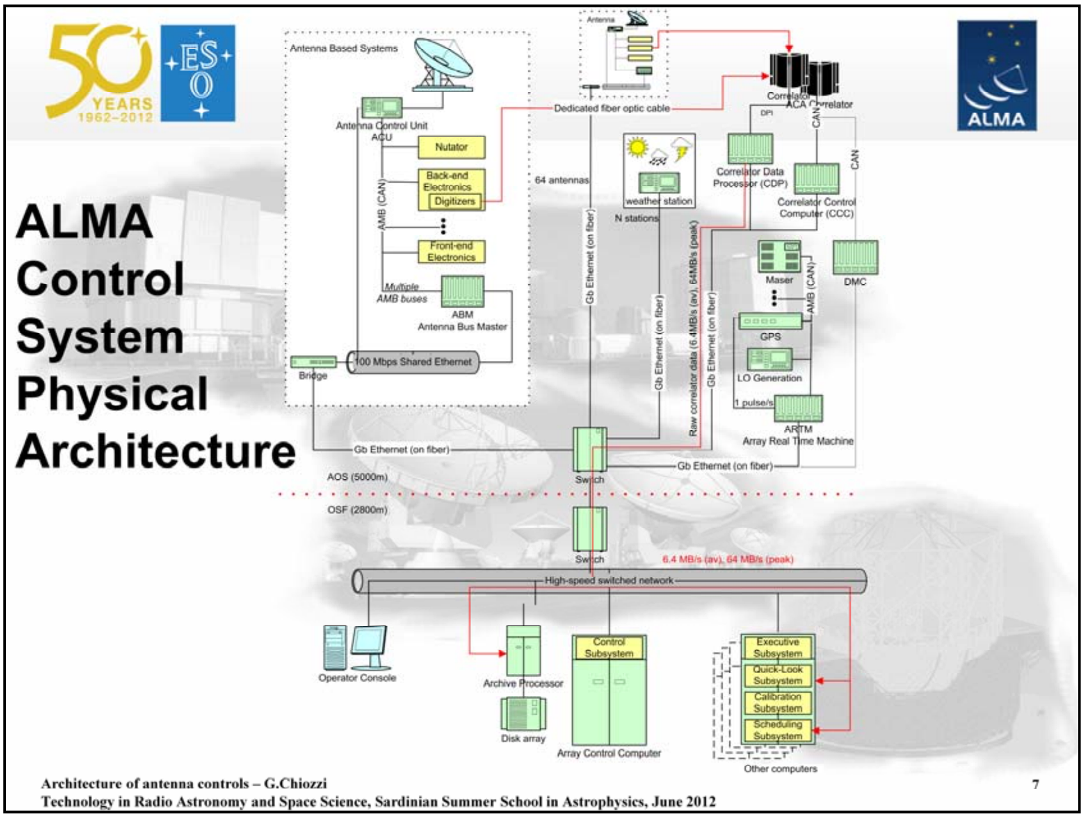
Maximum elevation angular acceleration: > 9 deg/ s2

• Axes must be able to achieve these rates simultaneously.

• Non-repeatable pointing errors are pointing errors that vary with time or are not-repeatable as a function of position. Are due to wind, temperature differences/changes, acceleration forces, encoder resolution, encoder errors, servo and drive errors, position update rate, bearing-non-repeatability and other similar sources.

• Non-repeatable pointing error for “absolute” pointing on the whole sky shall not exceed 2.0 arcsec RSS

• Non-repeatable “offset” pointing and tracking error shall not exceed 0.6 arcsec RSS



At each antenna, the Antenna Bus Master (ABM), a single computer, monitors and controls all the hardware devices in each antenna.

This includes the antenna servos, receivers, data samplers and power supplies.

This computer utilizes multiple ALMA Monitor Bus (AMB) networks to communicate with these devices. The AMB is a slight modification to the Controller Area Network (CAN) bus. The CAN bus is an industry standard, deterministic multi-drop serial connection that uses twisted pair cabling.

Much of the data from each antenna is collected via this ABM and routed via a gigabit Ethernet to the control system. However, the high-rate astronomical data collected by the receivers and digitized by the samplers is sent directly to the correlator via specialized fiber optic cables, completely outside the computer system. The DTS system modules, DTS Transmitter, DTS Receiver, and Fiber Optic Amplifier Demux cards (configured for each antenna/correlator-quadrant pair), make up this portion of the hardware system.

The correlator subsystem consists of a Correlator Control Computer (CCC) and a cluster of computers known as the Beowulf Correlator Data Processing Computer (CDP). The CDP itself is structured into four quadrants. The raw, correlated output data is routed directly to the archive and to the calibration and quick-look pipeline systems.

There are a number of weather stations connected to the control system.

In addition the ALMA system will deploy one other computer within 40m of central devices, the Array Real Time Machine (ARTM). The ARTM will monitor and control central devices such as the local oscillator generator, GPS and maser.

All of the above computers operate at the AOS at 5000m. They are diskless, but with flash memory disks, passively cooled, and boot and load their software over the network from central computers at the Operations Support Facility (OSF), at 2800m. These computers run a real-time operating system (and deterministic CAN buses).

The OSF contains the Array Control Computer (ACC) on which the central portions of the control system operate. Only the device-level portions of the control system run in the ABMs at each antenna and in the ARTM. The archive processor also operates at the OSF and is essential to the operation of ALMA.

Tasks of the Antenna Control System

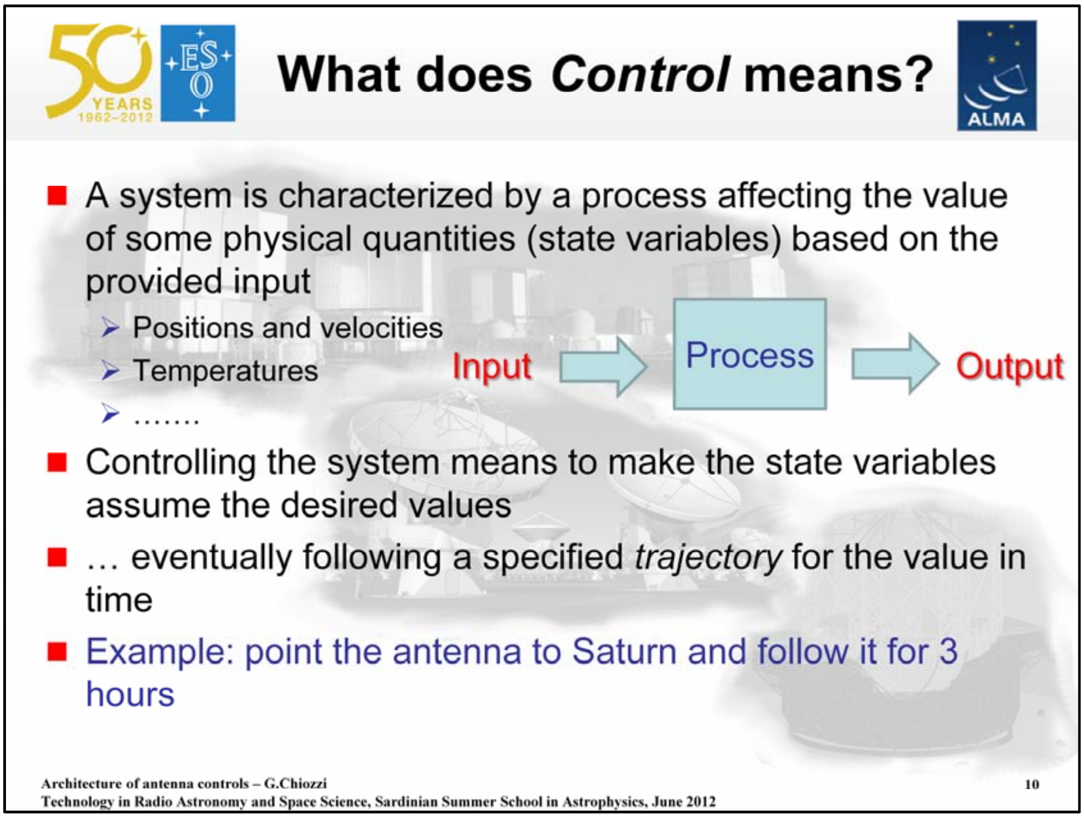
- Point the antenna/telescope at the desired target
- Keep it on target (tracking) following sky and object movements as well as implementing specific patterns in the sky
- Allow optimizing the configuration of the system to acquire the best data possible given the environmental conditions
- Allow the configuration of the devices to match the requirements of the specific observation
- Collect the “telemetry” information to analyze and diagnose the performance of the system.

Antenna controllable components

Main elements to be controlled:

- Az/El mount
  - Drives and encoders
  - Brakes
  - Pointing and tracking
  - Cable wraps
  - Locking pins
- Subreflector positioning
- Feed shutter
- Optical telescope
- Interlocks and switches for safety
- Power distribution and auxiliary
- Hydraulic systems
- Temperatures
- Metrology system
- Portable control unit

~1000 control and monitor points per antenna (~200 per second)



For more details on the control theory basics described in the following few slides,
You can look at this very good book (also mentioned in the references):

• Modern Control systems, R.C.Dorf R.H.Bishop

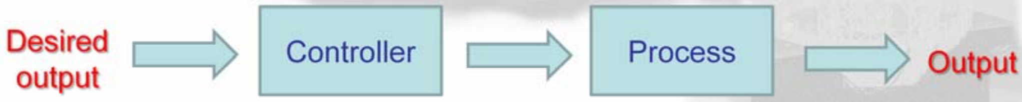
There is also an online course based on the contents of the book:

• http://www.calvin.edu/~pribeiro/courses/engr315/315_frames.html

Open-loop control

In order to control a system you need to know:

- What you want to achieve, i.e.
  desired value and behavior for the state variables (typically calculated based on a model of the system)
  - Example: The desired azimuth and elevation of the antenna to point to Saturn can be calculated using positional formulas and knowing time and location on earth.
- An open-loop control system uses a *controller* to obtain the desired response.



Desired response vs. output

Desired and actual output values are different since:

- Your formulas come from imperfect models. You cannot take everything into account.
  Example: Did you take into account precession effects?
- There are disturbances
  Example: Wind is shaking your antenna
- The system is imperfect
  Example: The sub-reflector is heavy and the quadrupod flexes
- The system is dynamic
  Example: The structure is heavy and inertia makes it overshoot the desired position

Closed-loop control

You are better off if you also know:

- How well the system is doing, i.e.
  measure/monitor the values of the state variables
  - Read from an encoder the actual azimuth and elevation
    or
  - Take an image and see if Saturn is in the center

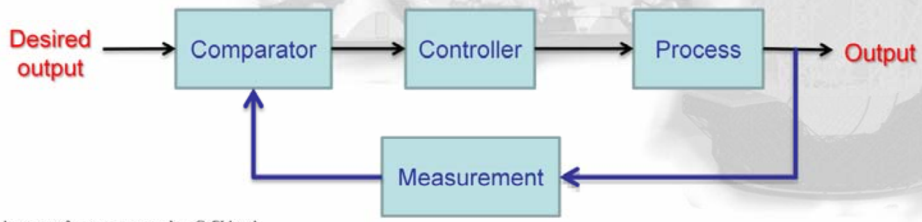
… since you improve the performance if you have a way to:

- feed back the measured values
- into the calculation of the desired values
- to correct the calculation and produce new input values that will bring the output values where you want

Closed-loop control

This is the basic principle of close-loop (or feedback) control

- A closed-loop control system compares actual and desired output and applies corrections to the *controller* behavior.
- Usually, the output, as measured by the sensor, is subtracted from the desired output in the *comparator*.
- That forms an error signal that the *controller* can use to control the plant.

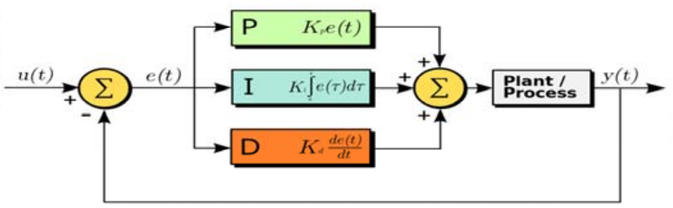


PID Controller

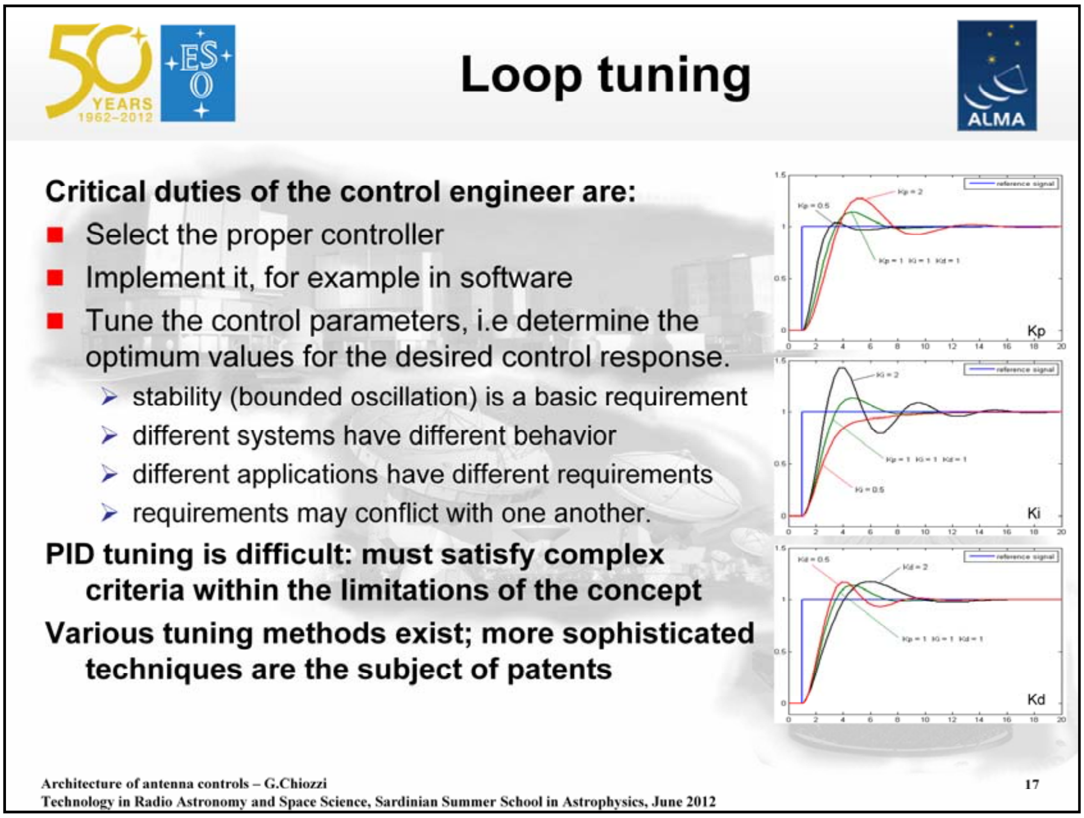
- The *controller* can be implemented in several different ways
- In the past, *controllers* have been implemented first mechanically and then with analog electronics
- Now they are mostly implemented using digital electronics and in software
- The most commonly used closed-loop *controller* is the PID

PID Controller -2

- PID stands for **proportional–integral–derivative**
- 3 separate constant parameters (called **three-term control):** the proportional, the integral and derivative values, denoted *P, I,* and *D.*
- euristically, these values can be interpreted in terms of time:
  - *P* depends on the *present* error,
  - *I* on the accumulation of *past* errors
  - *D* is a prediction of *future* errors, based on current rate of change.
- The weighted sum of these three actions is used to adjust the process via a control element such as the position of an axis or the power supplied to a motor.



$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$



A couple of notes:

1) Control System Engineers now how to design a control loop, how to model it using
tools like Matlab and simulink and how to tune them.

On the other hand, Software Engineers know how to efficiently implement the related algorithms
For a specific target platform (operating system and programming language).

The best constellation is that where a Control and a Software engineer can collaborate tightly in the design,
Implementation and commissioning of a control system.

2) An example of the need of using different control parameters or even different control algorithms
is the slewing and tracking behavior of an antenna.

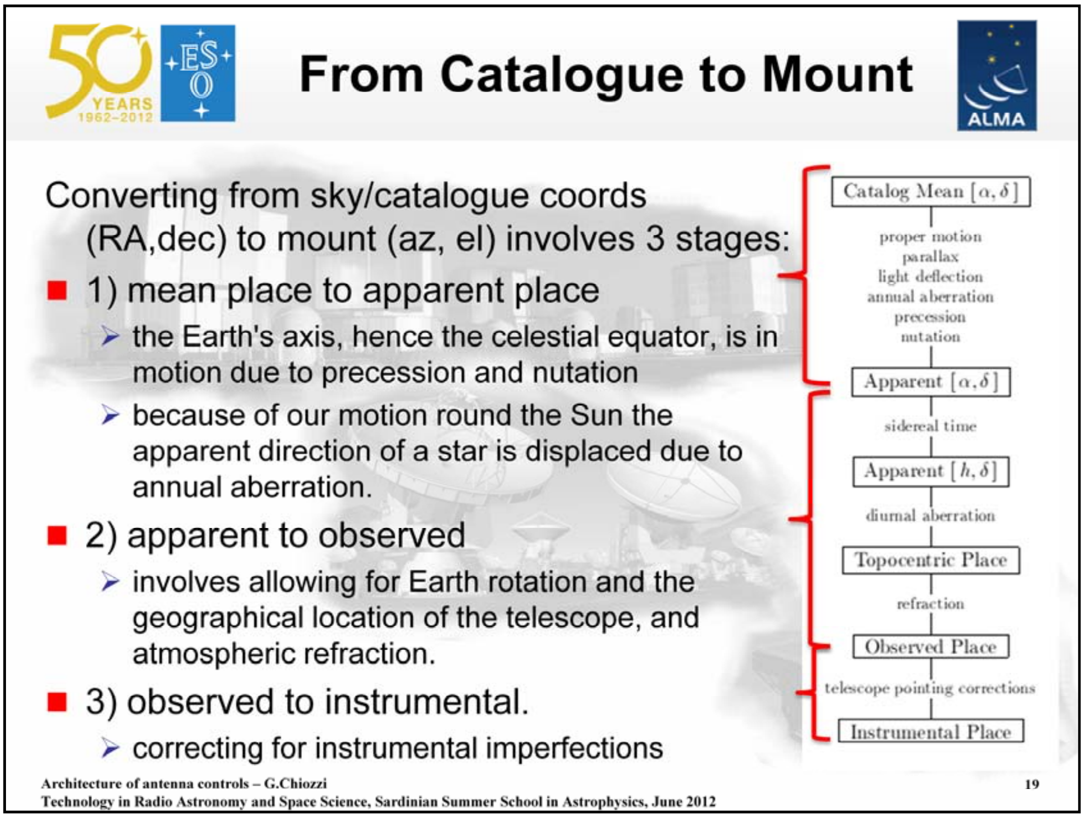
When slewing to a new target, the antenna has to move as fast as possible and we do not care about
A specific trajectory or wind rejection.

Once the target position has been reached, the antenna has to switch to a slow motion mode, following the
Trajectory of the source in the sky with high precision and reacting to friction and wind, among other disturbances.

This implies very different control parameters in the two mode and the need to smoothly switch
From one set to the other.

Pointing and tracking

- Primary task of the control system is to point the telescope to the target and keep it on target (tracking) following sky and object movements as well as implementing specific patterns in the sky
- Tracking converts (Ra, Dec) coordinates in real time in (Az,El) for the mount
- Tracking can be implemented in stages
- Pointing corrections are applied to correct for systematic errors
- The system has to minimize also dynamic effects introduced by mechanics and external factors, like wind.



More details on the conversion from (RA,dec) to (az,el) can be found in the referenced documents.

In particular:

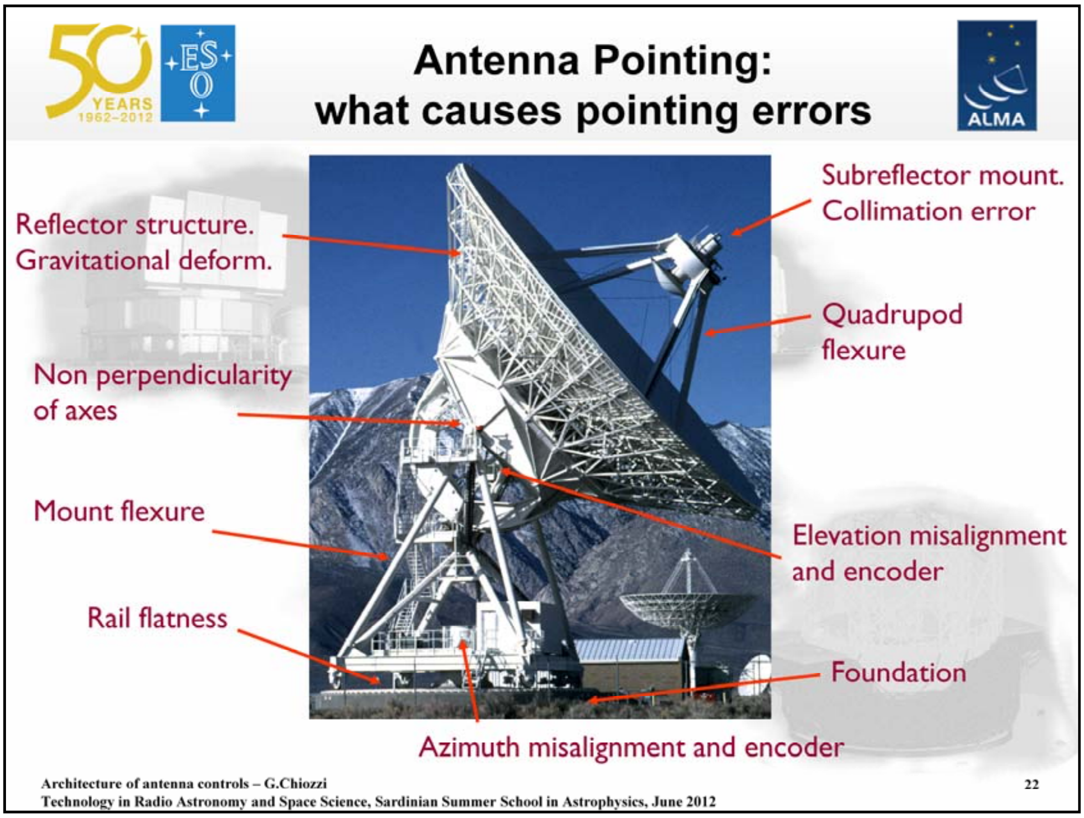
- Tpoint/SLA documentation: http://www.tpsoft.demon.co.uk/pointing.htm
- Standards of Fundamental Astronomy – AIU: http://www.iausofa.org/

From Catalogue to Mount - 2

- Steps 1 and 2 require
  - accurate positional astronomy calculations (like what provided by the SLA library)
  - Precise knowledge of time (GPS time or atomic clock)
  - Accurate measure of the position of the telescope on earth.
- In order to track the target, calculations have to be updated. Order of 10ms
- Some slow varying effects, computationally intensive, can be evaluated in a slower loop. Order of 1 second or less
- Update intervals depends on the required precision

From Catalogue to Mount - 3

- Step 3 requires an analysis of the imperfections of the telescope.
- Adjusting the system until it is as good as we can is not practical:
  - considerations of time and cost
  - may prove difficult to diagnose the deficiencies
- A more practical plan is to accept the imperfections and correct the star coordinates to take them into account.
- There are sophisticated libraries (like TPOINT) to:
  - Measure the errors
  - Calculate and apply corrections



The european AEM ALMA antenna, for example, include the implementation of a pointing model with terms to handle the following effects directly at Antenna Control Unit level:

- Azimuth encoder zero offset
- Collimation error of the electromagnetic axis
- Non-perpendicularity of mount azimuth and elevation axes
- Azimuth axis offset / misalignment northsouth
- Azimuth axis offset / misalignment eastwest
- Elevation encoder zero offset
- Gravitational flexure correction at the horizon

Pointing modeling

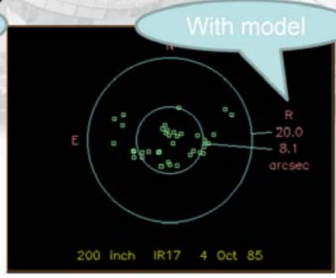
- Modeling all the distortions and irregularities in a telescope mount may seem an intractable problem.
- However, dramatic improvements can be achieved by correcting for a handful of effects common to all telescopes
  - ~6 six purely geometrical terms plus 2 or 3 flexures.
- The set of coefficient values and the formulas constitute the telescope's pointing model
- The pointing model gives the overall correction in each axis.
- The pointing model is essentially open-loop: Coefficients are estimated before observation and corresponding correction formulas applied to each (az, el)

Pointing model acquisition

- Applications like TPOINT allow to calculate the coefficients:
  - Accept lists of pointing observations specifying
    - (i) where the star really was and
    - (ii) where the telescope readouts said the star was.
  - Fit a user-specified pointing model (the desired list of coefficient names) to the observations, so that the coefficient values give the best possible match between star positions and corrected readouts.
  - Display in a variety of graphs formats the remaining pointing errors (the residuals). If the plots suggest that systematic errors remain, the operator can include additional terms in the model and try again.
- Ability and experience of the telescope control engineer are essential to identify the best set of coefficients

Radio Telescope pointing model acquisition

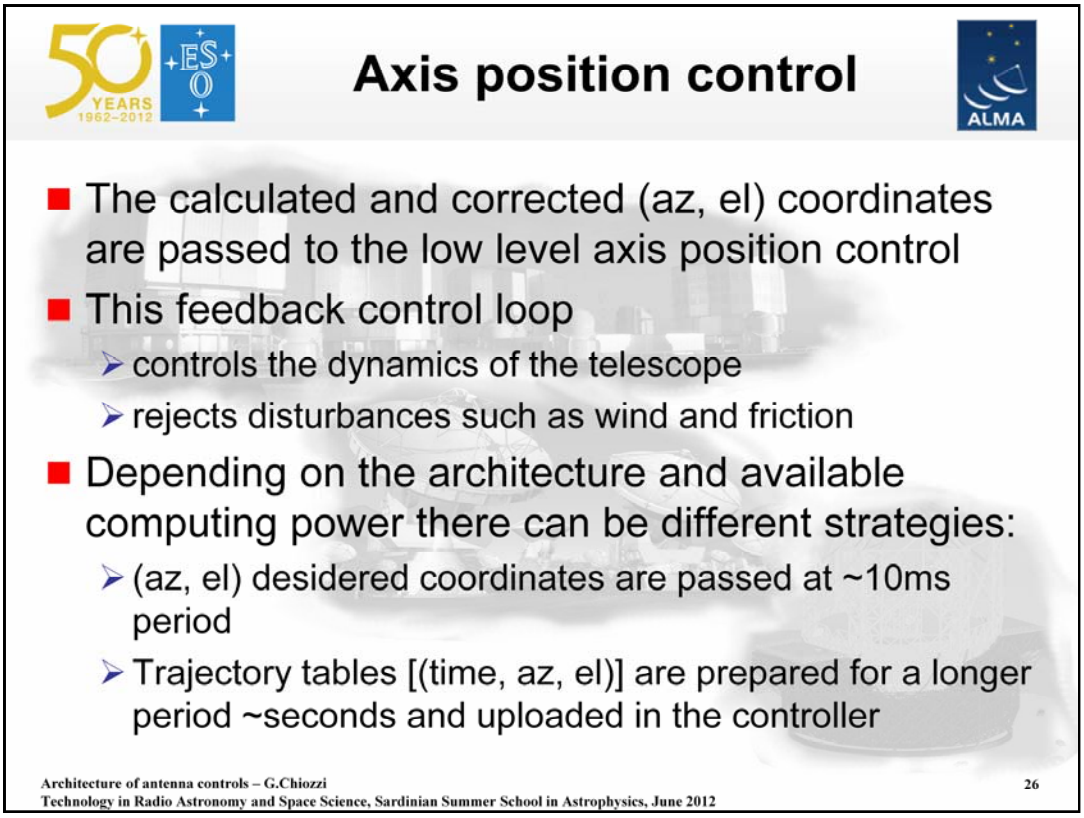
- The initial characterization is typically done using an optical telescope installed on the antenna
- Once there is a basic pointing model, it can be refined and repeated periodically using "pointing scan" observations (see the next presentation)
- Depending on the characteristics of the telescope, the pointing model might be more or less stable in time.



The 3rd picture shows a selection of TPOINT plots of the 200-inch data fitted with the basic 6-term model.

Runout is evident in both hour angle (top-left, east-west errors versus hour angle) and declination (top-center, declination errors versus declination). At this stage there also appears to be tube flexure (top-right, zenith distance errors versus zenith distance) and fork flexure (center, declination errors versus hour angle) but it turns out these go away when the runouts are corrected.

The other plots are east-west errors against declination (center-left), h/d nonperpendicularity versus hour angle (center-right), the scatter diagram (bottom-left), the error distributions (bottom-center) and the map of error vectors on the sky (bottom-right).



The vendor of each ALMA antenna has developed a numerical model trying to comprehend all relevant elements that may influence the actual

dynamic behavior of the antenna:

· structural dynamic response

· effect of wind disturbances

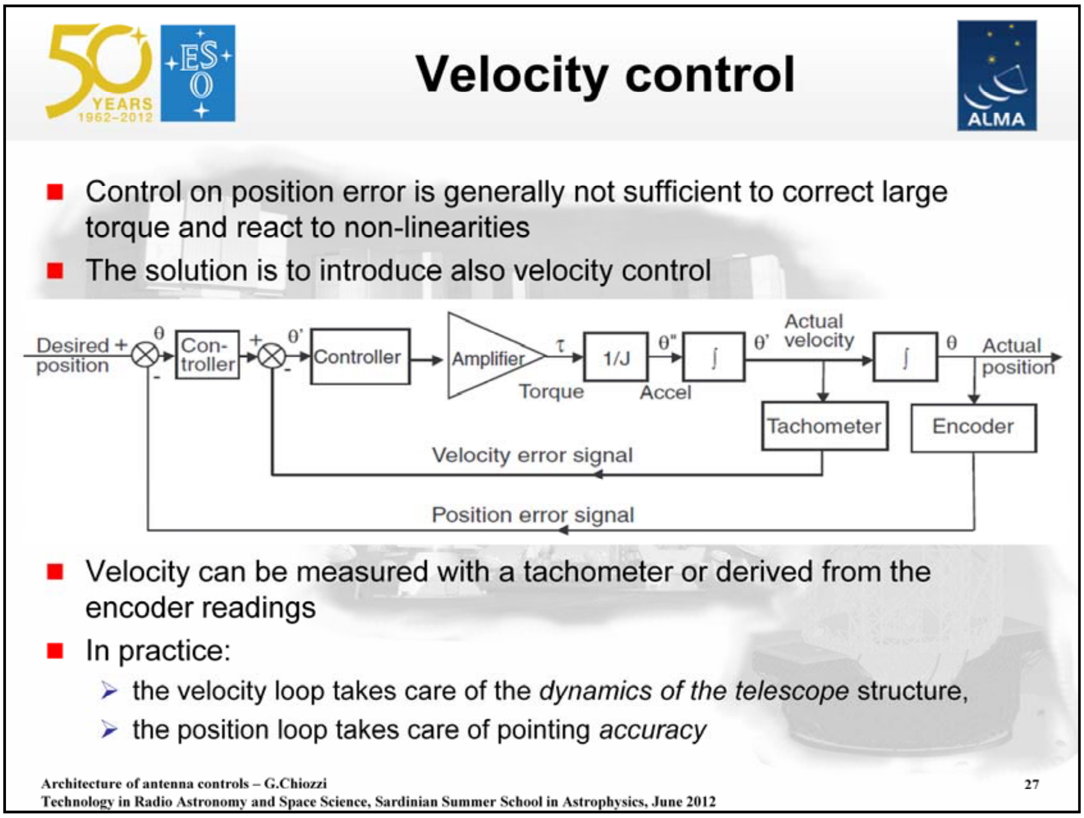
· non-linear phenomena, like 'stick and slip' of bearings and roller guides

· motor and motor drivers bandwidth limitations

· motors torque noises

· encoder system noise and quantization effects

These are documented in details in the respective design documents.



The velocity loop tightly controls the dynamics of the telescope structure and rejects disturbances such as wind and friction.

The loop is usually implemented with a PI controller to maximize responsiveness, combined with a filter to avoid exciting resonance frequencies in the telescope structure.

In practice, it is only possible to extend the bandwidth up to about 60% of the lowest locked rotor frequency of each telescope axis.

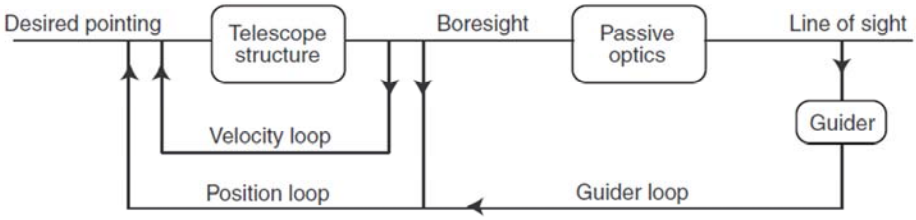
With velocity control solved, the position loop’s task is reduced to maintaining zero tracking error and to handling large changes in the desired position.

During tracking (small excursions), a PI controller is used to maximize responsiveness.

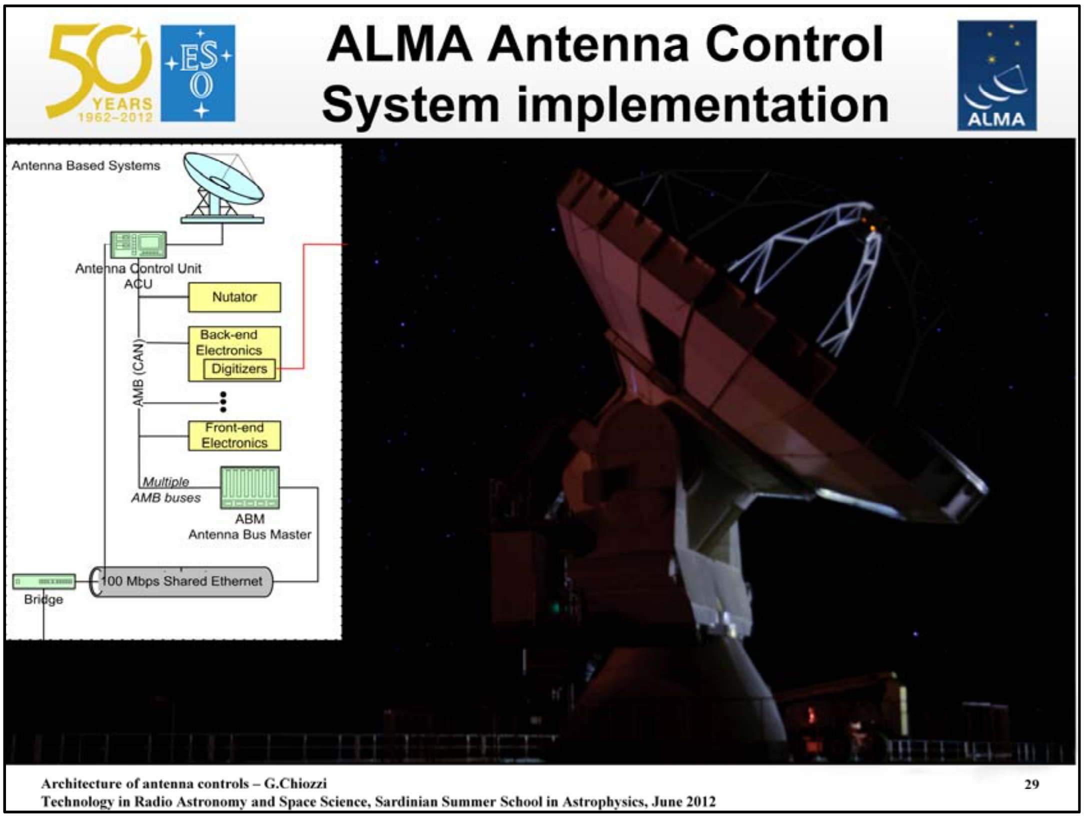
Large steps in position commands (large offset requests, repointing) are handled with a special algorithm.

Guiding

- Optical telescopes can actively correct errors by *auto-guide:*
  - A guiding camera points to a brigh *guide star* inside the telescope field, but outside the science field
  - Centroids are calculated
  - Error with respect to theoretical position is sent to the position loop as additional correction



- Radio telescopes **cannot** do this, but the execution of calibration and pointing patterns during the observations can be used in a similar (less efficient) way



After having presented the general concepts, we look into possible implementation solutions, taking the ALMA Control System as an example.

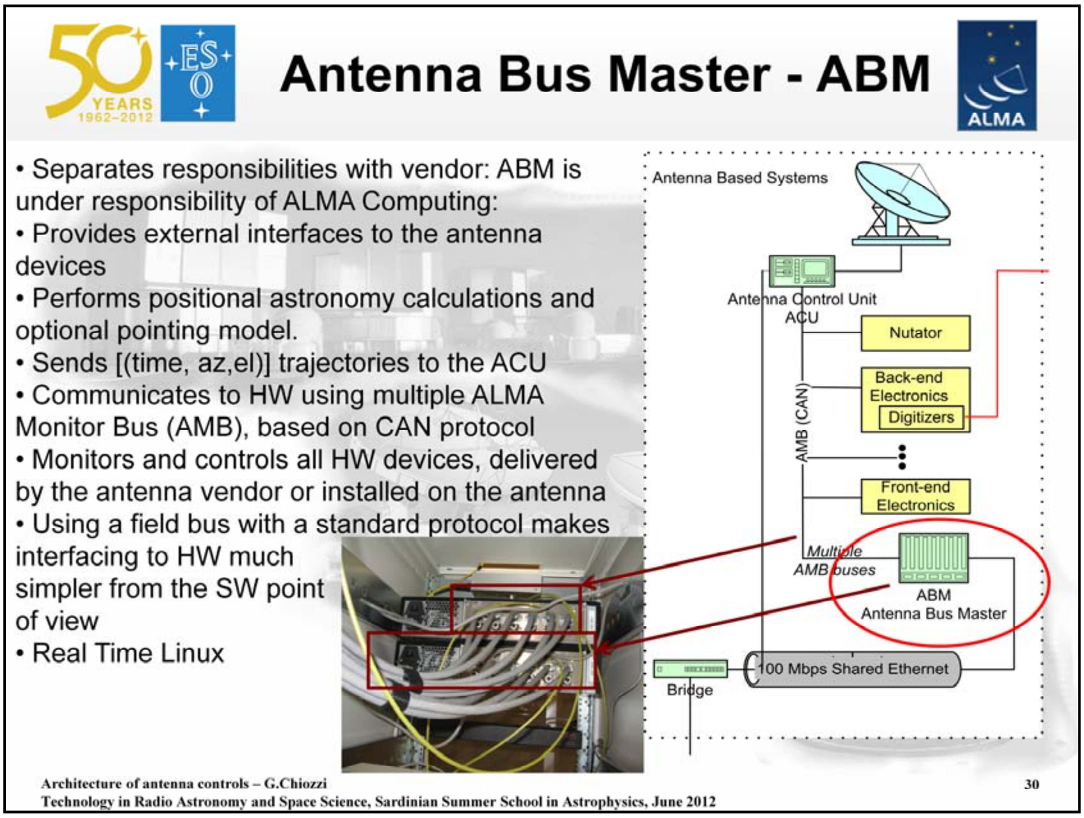
Here starts the second part of the talk, with an analysis of the design of the ALMA Antenna Control system.

Looking at the schema above, you can see that the interface with the external world is responsibility of the Antenna Bus Master computer.

On the other hand, control of the antenna low level functions is responsibility of the Antenna Control Unit computer.

The existence of the two computers is the result of a specific design decision with the purpose of clearly identify the responsibilities of the vendor.

This will be explained in the coming slides.



The Antenna Bus Master (ABM) computer has been introduced in ALMA to clearly separate the responsibilities of the antenna vendor from the responsibilities of the ALMA computing team.

The ABM is under the complete responsibility of ALMA Computing and it:

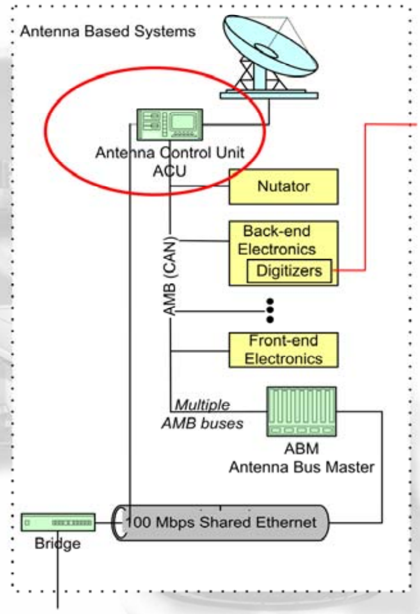
• Provides interfaces to allow all other subsystems to get access to the devices installed on the antennas using the standard ALMA infrastructure

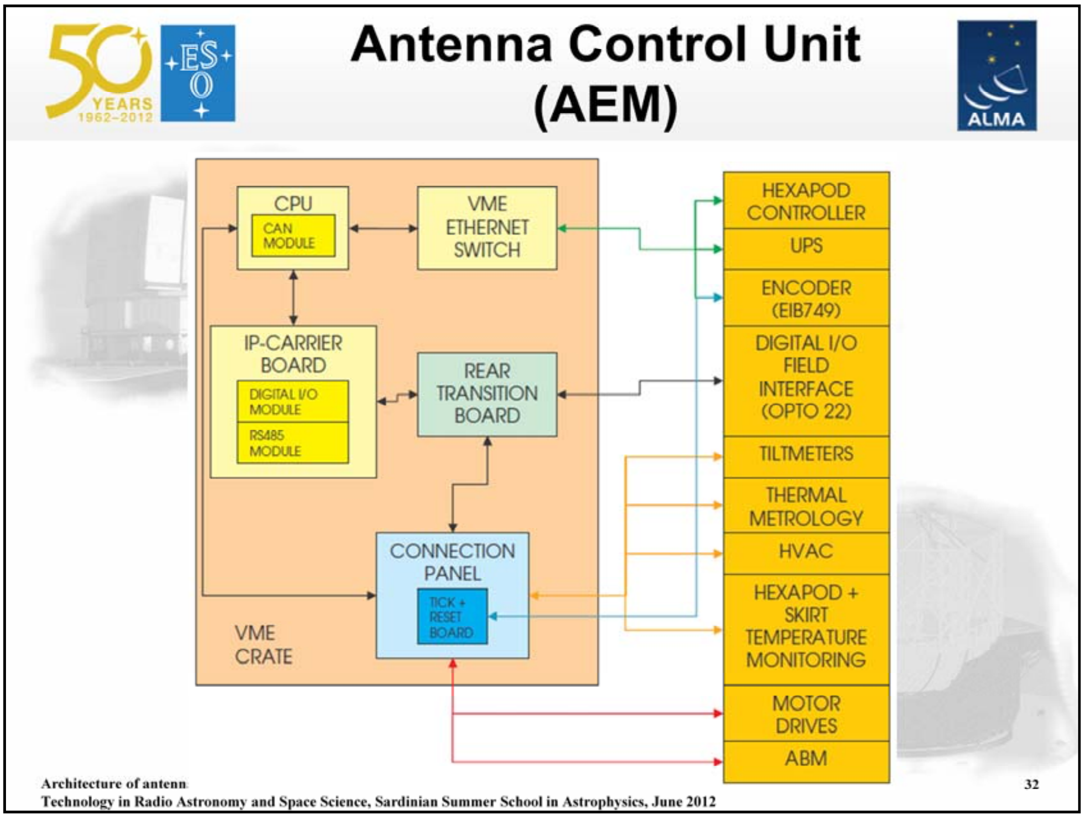
• Interfaces to the hardware on the antennas using a field bus technology (AMB, based on the CAN standard) to make access uniform and therefore easier.

• Real Time Linux is used to implement the communication protocol and time synchronization functions, so that monitoring and control are deterministic.

Antenna Control Unit - ACU

- The ACU is under responsibility of the antenna vendor. A black box.
- Does not contain knowledge of astronomical concepts
- Interfaces to the ABM via CAN bus
- Controls directly the antenna hardware as designed by the vendor: PLCs, digital/analog I/O, serial protocols, other industrial buses
- Runs the antenna position control loops in real time (using Real Time Linux)
- Optional pointing modeling
- Is synchronized with the 48 ms Timing Event





The ALMA ACU is based on a VME crate, placed inside a cabinet on the Azimuth platform.

In the AEM implementation, the ALMA ACU is based on a GE Fanuc Intel Pentium 4 VME single board computer running Real Time Linux.

For the CAN communication interface, the Tews Technologies PMC 901 Extended CAN board has been chosen.

This board embeds six completely independent CAN 2.0 controller based on the standard Intel 82527

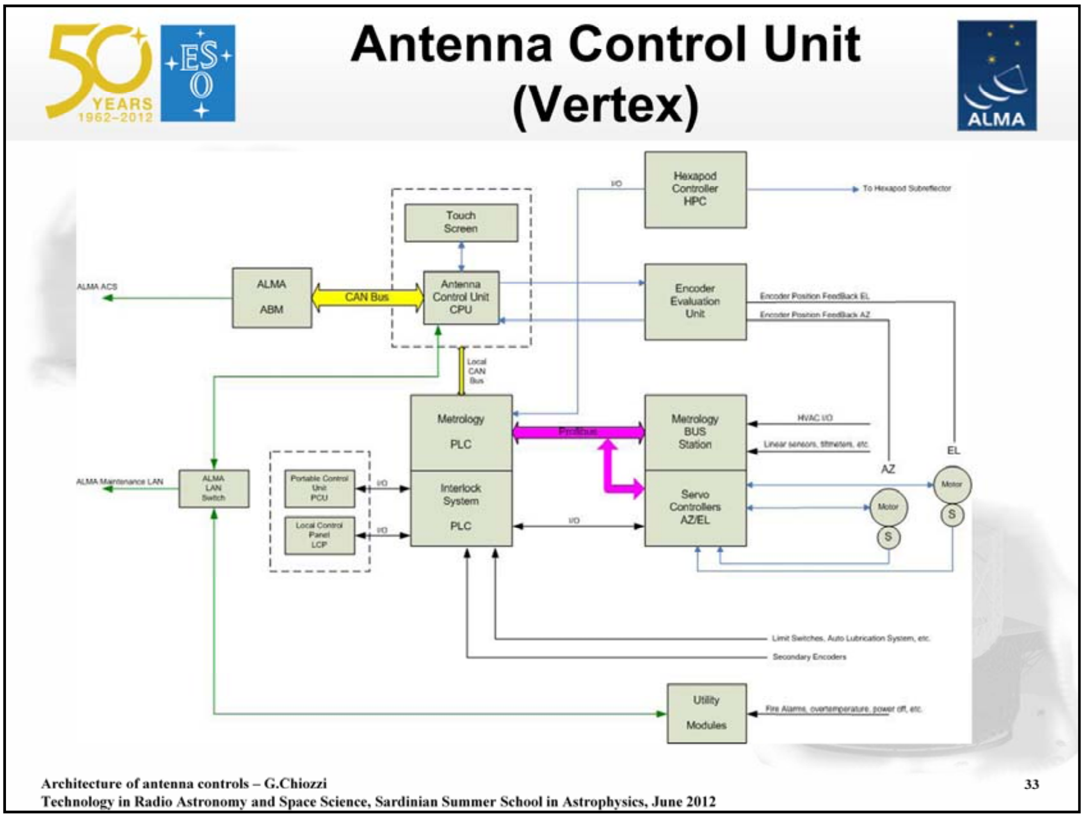
Five of the six available CAN ports are used:

· 1x CAN line for ALMA ABM, for antenna operation in remote mode

· 4x CAN lines for motor drivers (torque commands, encoder positions and Hall sensors).

For all communication and I/O boards the IP standard was chosen. The IP standard is currently the most common and supported form for VME expansion boards. Even not being the most performing standard currently available, it is a good tradeoff between I/O density and communication performance.

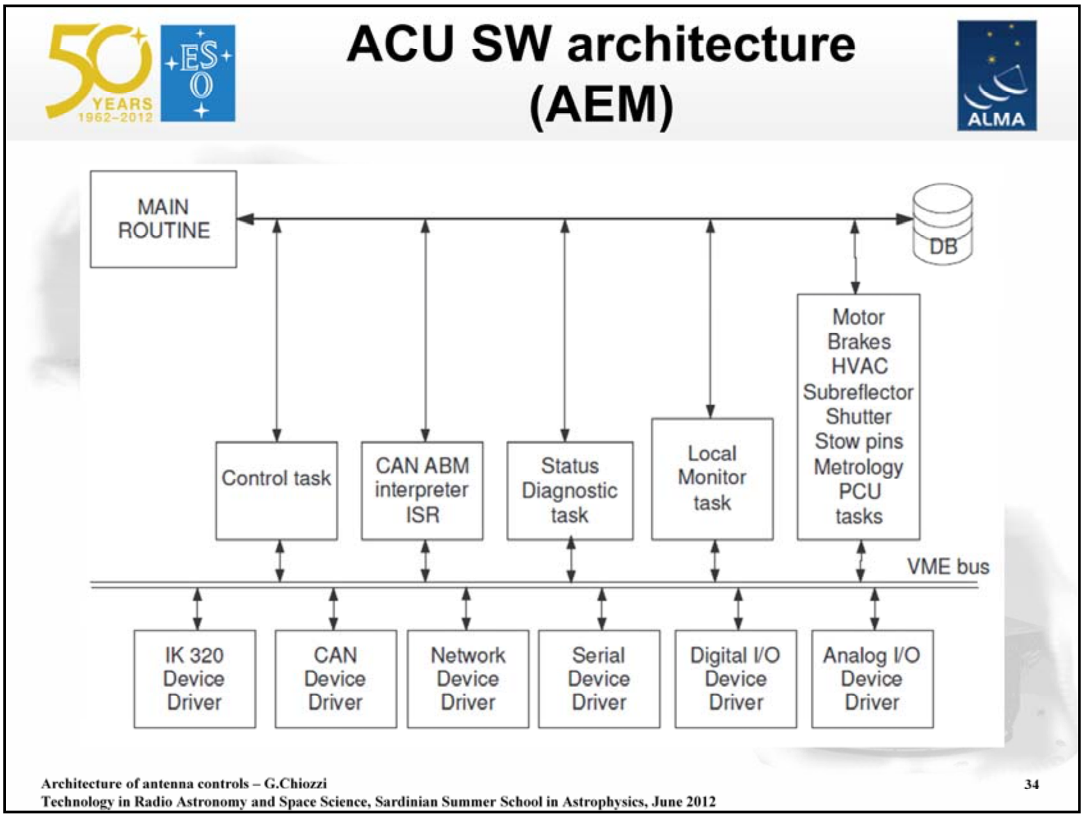
The motor driving system is based on the Phase TW3 vector motor drives. The control loop runs in the ACU, that sends at every control cycle a torque command to the drivers. Additionally, the absolute angular position information is passed to the drivers for proper electrical synchronization



Vertex design is different but provides mostly the same interfaces and satisfies the same requirements

The can bus is used only to interface the ACU CPU with the ABM.

Some of the control functions are implemented using industrial PLCs, interfaced with metrology and servo controllers using profibus



This diagram shows the software architecture in the AEM ACU.

The Control task is activated as soon as the EIB749 boards have finished the encoder reading job and have sent the encoder data via UDP. When activated, this routine performs the following actions:

· implements the servo control loop algorithm,

· checks some safety conditions,

· interprets some ABM commands

· computes the trajectory for next step

· then the routine returns to wait for the next semaphore setting.

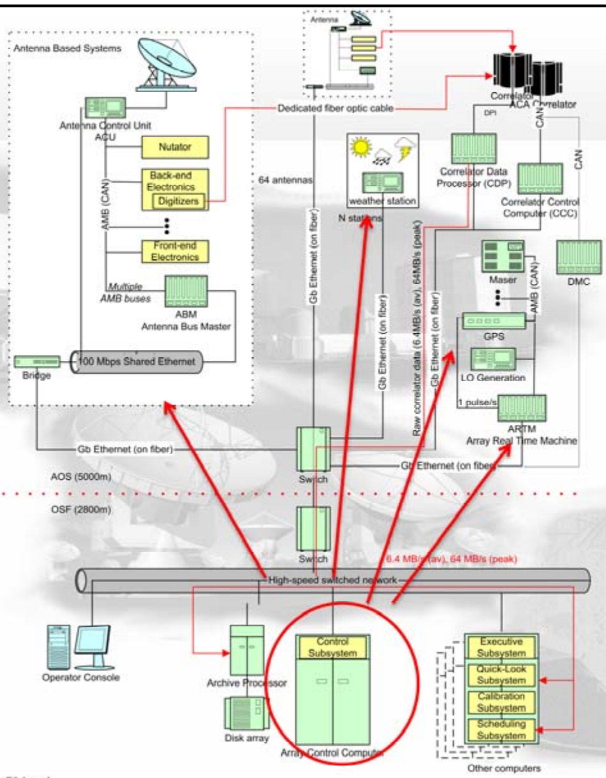
The encoder latching is activated via hardware by the tick + reset board.

Status, Motor, Brakes, HVAC, Subreflector, Shutter, Stow pins, Metrology handling task and Local, Diagnostic monitor tasks are called by the operating system according to a predefined timing sequence.

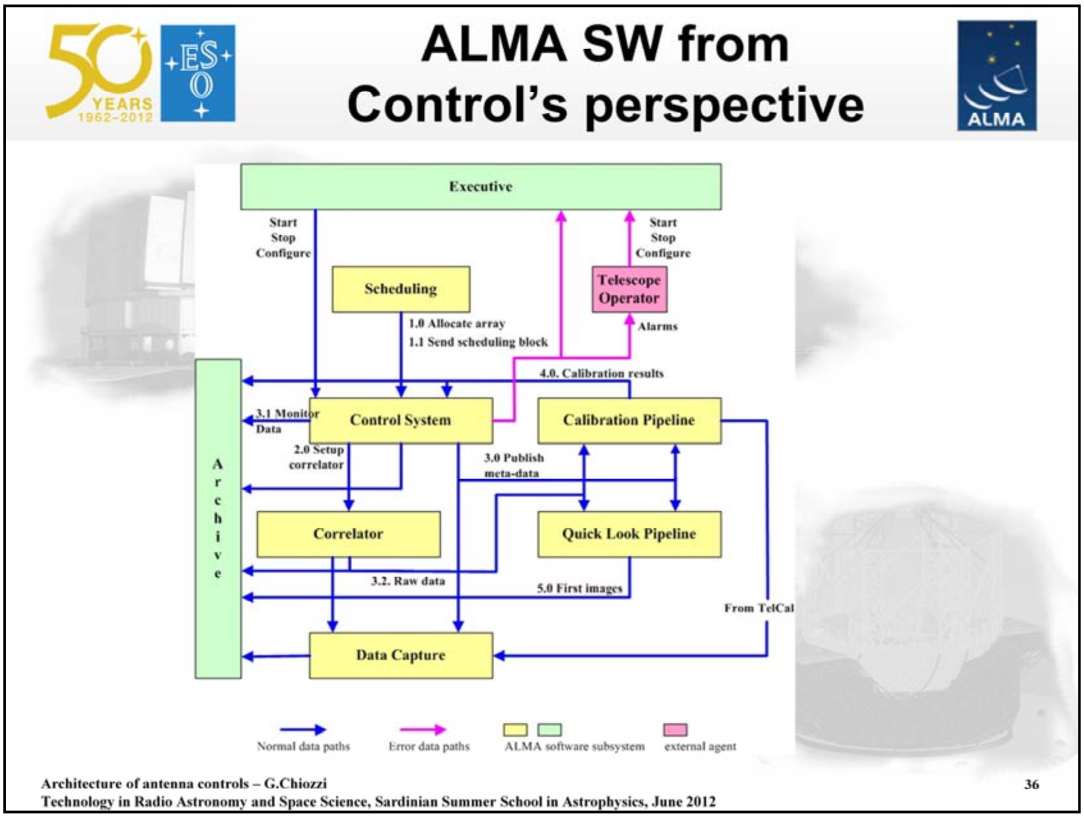
The ABM interpreter task is activated through the CAN interrupt generated when new data is available.

The initialization routine allocates the DB memory, initializes all the tasks, and prepares the system for receiving the external commands.

ALMA High Level Control System



Coordination and interface to other subsystems



This diagram shows the interactions between the different subsystems running at the ALMA observatory.

The Executive control and monitors the status of the whole system. It starts the Control system by telling it to initialize itself.

The Control system is responsible for initializing and configuring all hardware devices within the system.

A startup configuration is read from the Telescope Monitor and Configuration Database (TMCDB) in the Archive. This is a list of hardware that is expected to be on-line.

At this point the Control system discovers what hardware is actually available for it to manage.

The Control system also determines whether correlators, calibration and archive systems are available at this point.

Normal scientific observations are initiated when the Executive tells the Scheduling system to begin scheduling work.

Work gets to the control system in the form of scheduling blocks.

These are stored in the archive as parts of observing projects.

The Control subsystem executes the scheduling block commanding the antennas, correlator, and other hardware.

The result is that raw data from the Correlator and meta-data from the Control subsystem are made available to the Calibration and Quick-look pipelines.

This raw-data and meta-data are also stored in the archive.

Subsequent parts of the system analyze the data and inform the PI of its availability.

It is important to understand the relationship between the Scheduling subsystem and the Control subsystem.

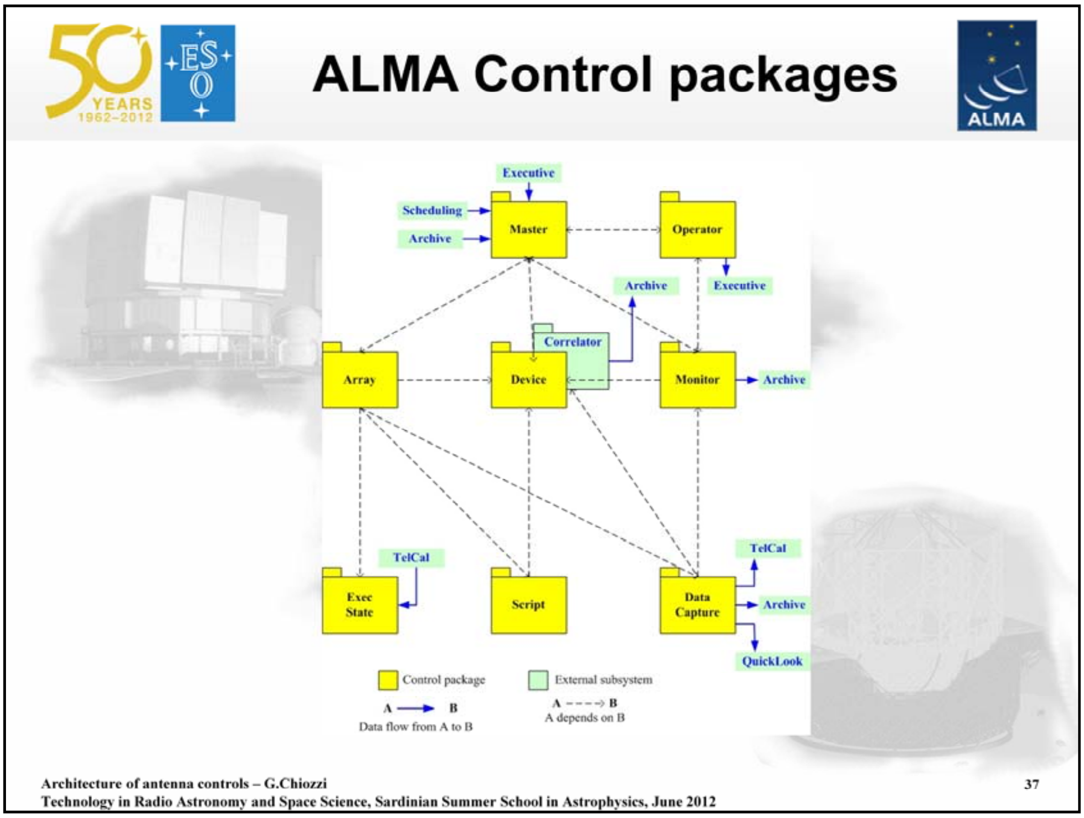
Scheduling operates in near real-time and manages the execution of scientific projects.

Scheduling manages the creation and destruction of arrays and the allocation of antennas to arrays.

In Control, arrays are created and are operated in one of two modes:

Automatic – "normal" operating mode; executes scheduling blocks with pre-defined observing scripts, sends data to Data Capture, and stores data in the archive.

Manual – an interactive mode intended for diagnostics, testing, and special development; execution is via command line or scripts.



Device
This package allows a SW representation of the devices in the system and spans a broad range of concepts and functionality: from a low-level software component controlling a power supply to an antenna, viewed as a collection of devices working together as a unit.
Each device implements a state model (start, configure, initialize, operational, error, and stop) with well-defined transitions between these states. Each device has a parent module.
Every device has a public interface and is accessible over the distributed network.

There are basically two types of software components associated with devices: low-level device-drivers that directly interact with a specific hardware and higher-level software components that are aggregates of these lower-level components designed to function as a unit.

The ALMA telescope has more than 50 types of devices and between 3,000 and 4,000 instances of those devices. The vast majority of these are monitored and controlled by a software module.

Higher-level devices are crafted as collections of low-level devices that function as a unit for some purpose. These may be thought of as "logical" devices and are frequently created on-the-fly as needed.

One obvious case is an antenna. From this perspective, an antenna is a logical device containing, for example, a mount, optical telescope, holography receiver, as well as other kinds of front-end receivers, and a whole host of back-end devices and a DTS system that connects it to a correlator.

Monitor
The Monitor is started by Control's Master and runs continuously and independently of all other packages in the control system; it stops only on command from the Master.
The monitor package is responsible for collecting data in such a way as to reduce the data storage transaction rate into the archive. The fully operational ALMA telescope will have approximately 4,000 devices, each continuously generating substantial amounts of monitor data. The sampling of monitor points can be at rates of up to ~20Hz. Higher sampling rates on selected hardware will be supported for limited periods of time.

Operator
The Operator package allows for multiple displays to multiple telescope operators, separating the physical presentation from the logical structure of the information being displayed.
The Operator package provides mechanisms for displaying all monitor data, including interpretations, summaries and inferences. The state of any device can be displayed, as well as aggregates such as individual antennas or sub-arrays.
The package also displays all alarms on a high priority basis.
The Operator GUI must have a look and feel that is consistent with other subsystems that interact with the telescope operator. In most cases the GUIs in this package take the form of "plug-ins" to the operator's console that is maintained by the Executive subsystem.

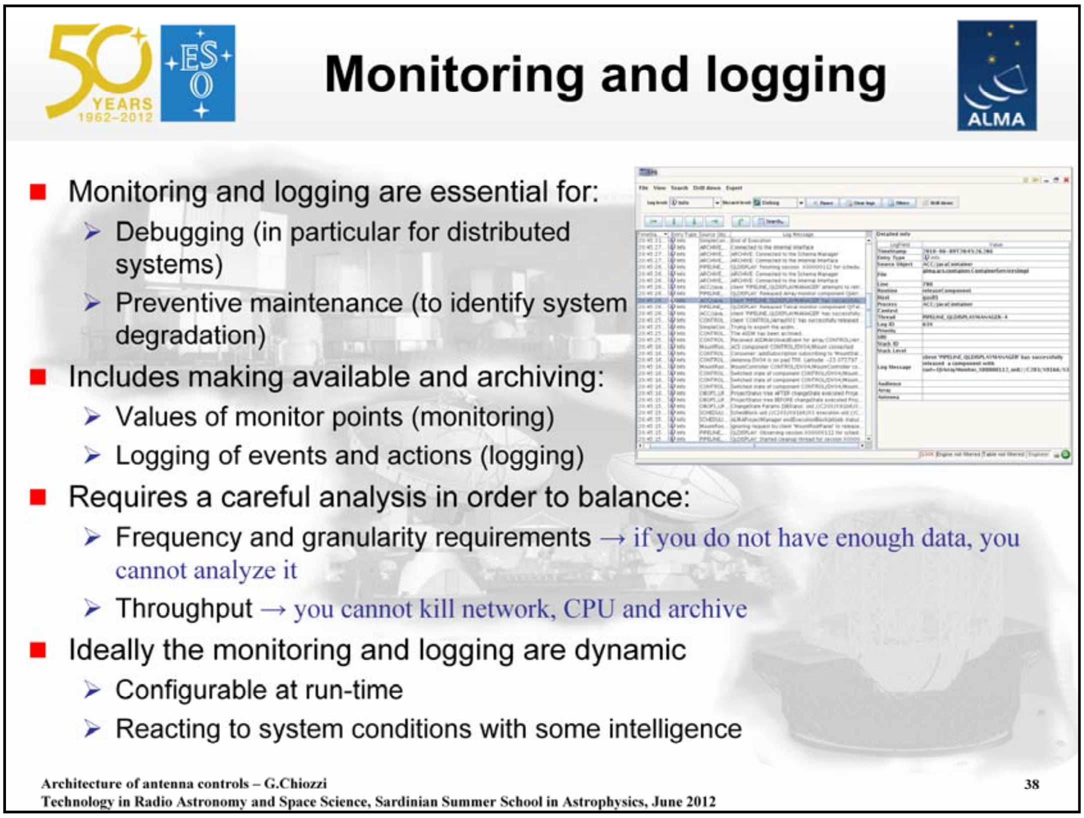
Array
Arrays operate in one of two modes: **automatic** (the "normal" operating mode in which scheduling blocks with pre-defined observing scripts are executed, data is sent to Data Capture, and stored in the archive) and **manual** (an interactive mode intended for diagnostics, testing, and special development; execution is via command line or scripts).
Antennas in manual mode are not available to Scheduling and are exclusively handled by Control. One or more antennas may be operational in a Manual Array, which operates under the control of a staff person: a software or hardware engineer or staff astronomer.
A single instantiation of an Array object controls one array. In general, the Control Master controls many Array objects; each of these functions independently and concurrently.
The Array package has the following responsibilities:
- Supervise the execution of a scheduling block (if in automatic mode),
- Directly execute commands (if in manual mode),
- Monitor all antennas allocated to the array and handle any errors in hardware,
- Handle any errors in the execution of scripts,
- Report to the Operator the current status of the array,
- Ensure that the time allowed for scheduling blocks or manual mode does not exceed its maximum allocated time.

Script Executor
The Script Executor contains an interpreter that executes scripts.
The Script Executor implements the Control Command Language (CCL)
This interpreter has access to a library of commands that control ALMA devices.
Therefore, when scripts that contain these commands are executed, the ALMA devices are commanded.
The CCL includes a library of standard astronomical functions, as well as the CASA measures and quanta libraries (via a Python wrapper), which give complete reference system conversions.
The Script Executor has the following responsibilities:
- Define the CCL,
- Maintain a library of standard observing modes,
- Provide high (science) level functions for controlling the array,
- Provide low level functions for more detailed access to the hardware,
- Decompose all these functions into device specific commands.

Execution State Model
The Execution State Model package has the following responsibilities:
- Maintains a library of data representing the current state of the pipeline calibration results, and data quality information,
- Makes this library available to the Script Executor in a form that is easily accessible by the script interpreter,
- Receives data from the calibration pipeline that represents the results of calibrations being carried out on current observations,

Data Capture
The Data Capture package translates the data that is being generated by the on-line subsystems i.e., Control, Correlator and Telescope Calibration and puts it into the archive in a format suitable for the offline subsystems.
Each of the on-line subsystems sends information to the data capture component as it is produced. The Data Capture package buffers and reorders this information, generating indices and derived quantities to ensure that what it writes to the archive is complete and suitable for further processing by the data reduction subsystems. The output data format is the ALMA Science Data Model (ASDM).

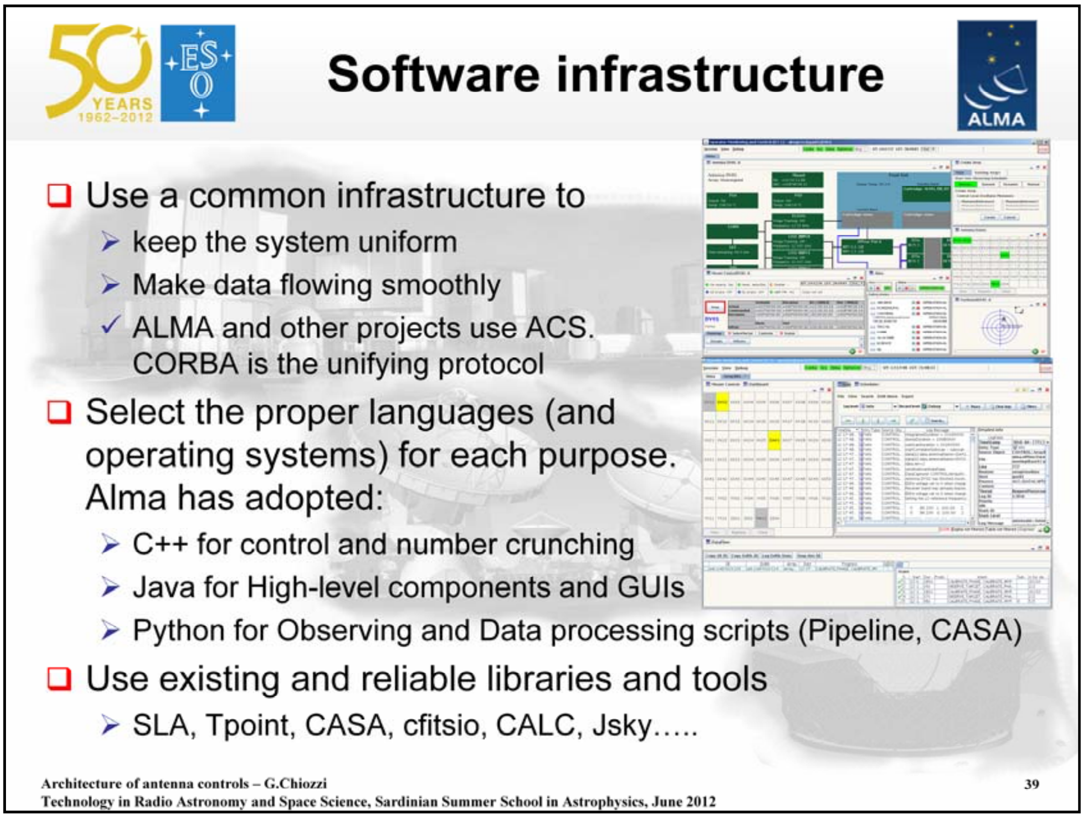
Master
The Master Controller is responsible for the following activities:
- Initializing and configuring all devices,
- Stopping all devices gracefully,
- Initiating all tasks within the subsystem, such as monitoring, that must run continuously,
- Making certain that continuously running tasks are in fact running and logging any errors detected,
- Stopping all tasks within the subsystem,
- Creating arrays from existing idle antennas in either manual or automatic mode,
- Ensuring that arrays can operate independently,
- Destroying arrays (antennas are marked idle),
- Handling alarms and dealing with them appropriately, including the possibility of shutting down any device.



Centralized monitoring and logging are essential services for the operation of a distributed system.

They are also probably the most important debugging tool for a distributed and concurrent system.

Using a source code debugger, it is in fact impossible to debug concurrent issues, because break points and function stepping heavily affect concurrency.

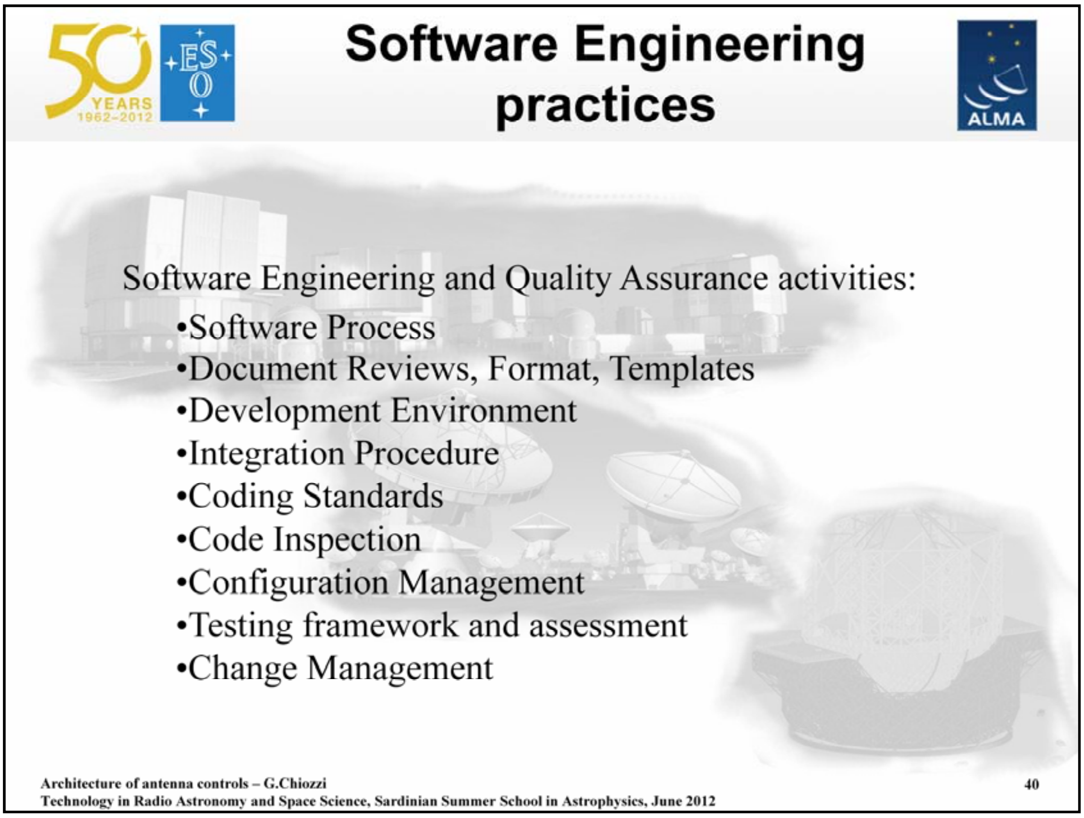


It is strongly suggested to have a common software infrastructure across the observatory, to allow a seamless integration of the different subsystems involved in the end-to-end data flow.

It is in particular very important to have

• a common inter-process communication protocol, so that all processes can easily exchange commands

• integrated monitoring and logging systems, so that all monitor point values and events in the system go in a single repository from where they can be analyzed.



Adopting Software Engineering practices can help substantially in keeping the development process under control.

Balancing the cost of the overhead introduced with the complexity of the project and the benefits that can be reached will dictate up to which point it makes sense to push for formal practices.

But in any case it is counter productive to simply state rules on paper and ask people to follow them.

It is essential to provide tools and support so that the adoption of the practices and their verification is transparent or becomes second nature.

For example, the choice of technologies can have an impact on the software engineering practices. As an example we can consider the adoption of tools like LabVIEW. These tools do not integrate naturally with a traditional source code configuration management system, because of the structure of the projects, containing a lot of binary information. The E-ELT SW Engineering team is therefore analyzing now what is the best strategy to handle LabVIEW artefacts.

Conclusions

- Implementing a Telescope Control system is a complex engineering task.
- It requires tight cooperation between
  - Control engineers
  - Electronics engineers
  - Software engineers
- It requires Astronomy and Telescope domain specific knowledge and therefore industrial outsourcing requires careful choices

References

- Design and construction of large optical telescopes, P.I.Bely, Springer, 2003
- Telescope control, M.Trueblood R.Merle Genet, Willmann-Bell.1997
- Modern Control systems, R.C.Dorf R.H.Bishop, Addison-Wesley
- Tpoint/SLA documentation: http://www.tpsoft.demon.co.uk/pointing.htm
- Standards of Fundamental Astronomy – AIU: http://www.iausofa.org/
- Requirements and design documents of projects. Typically available on the web.

Questions?