DISCOS: A common control software for the SRT and the other Italian radio telescopes

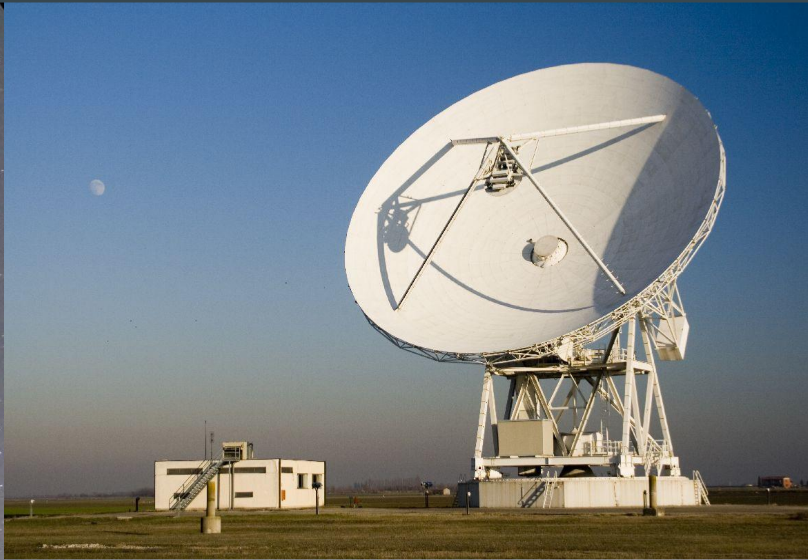
Sergio Poppi
on behalf of the DISCOS team

DISCOS Team

- **Carlo Migoni (INAF):** core developer, VLBI integration.
- **Andrea Orlati (INAF)**: team leader - project manager - core developer.
- **Marco Buttu (INAF):** core developer, test driven development.
- **Marco Bartolini (INAF):** core developer, continuous integration.
- **Simona Righini (INAF):** astronomical advisor, observations, documentation.
- **Antonietta Fara (INAF):** system administrator.
- **Sergio Poppi (INAF):** core developer, astronomical advisor, observations.

Project History

- 2004 Development SRT Control Software - NURAGHE started
  - team: Giuseppe Maccaferri, Andrea Orlati, Francesco Palagi, Carlo Migoni, Matteo Murgia, Francesco Schillirò (GAI SOFTWARE - SRT)
- Goal:
  - Provide the Sardinia Radio Telescope of control software with enhanced performances.
  - Build a common infrastructure for the three radio telescopes.
- 2007 ESCS Enhanced Single-dish Control System (Medicina and Noto)
  - team: GAI SOFTWARE + Simona Righini, Rashmi Verma, P.Libardi
- 2015 DISCOS: unifies the three development lines.


Sergio Poppi©



Telescopes Configurations

	SRT	Medicina	Noto
Main mirror	64 m	32 m	32 m
Optical configuration	Gregorian	Cassegrain	Cassegrain
Mount	Altazimuthal, fully steerable 12 motors + cable wrap	Altazimuthal, fully steerable 4 motors	Altazimuthal, fully steerable 4 motors
Antenna Control Unit (main servo system)	Beckhoff PLC ethernet vendor protocol	VxWorks based PC ethernet vendor protocol	VxWorks based PC ethernet vendor protocol
Primary Focus	three degrees of freedom INAF defined protocol	three degrees of freedom INAF defines protocol	
Secondary Focus	six degrees of freedom ethernet INAF protocol	five degrees of freedom ethernet INAF protocol	five degrees of freedom RS232 vendor protocol
Active Surface	1008 aluminium panels 1116 actuators rs485/ethernet vendor protocol	not available	240 aluminium panels 244 actuators rs232 vendor protocol

Telescopes Configurations

	SRT	Medicina	Noto
Main mirror	64 m	32 m	32 m
Receievers*	0.305-0.410 1.3-1.8 5.7-7.7 18.0-26.0, 7 feeds GPIB and ethernet INAF protocol	1.35-1.45 1.595-1.715 2.2-2.36 4.30-5.80 5.90-7.10 8.18-8.98 18.0-26.0, 2 feeds GPIB and ethernet and RS232 various protocols	0.317-0.320 1.40-1.72 2.20-2.36 4.70-5.05 8.18-8.58 22.18-22.46 39.0-43.3 GPIB and RS232 various protocols
Backends*	TotalPower (continuum) 0.1-2.1, 1-1000 ms, 14 inputs XARCOS (spectro-polarimetry) 0.0005-0.125, 10 s, 2048 bins, 14 inputs Roach(spectro-polarimetry) 0.512, 10-1000 ms, 8192 bins, up to 14 inputs DFB3(pulsar) 1.024, 1-4000 ms, 8192 bins, 4 inputs DBBC	TotalPower (continuum) 0.1-2.1, 1-1000 ms, 4 inputs XARCOS (spectro-polarimetry) 0.0005-0.125, 10 s, 2048 bins, 14 inputs	TotalPower (continuum) 0.1-2.1, 1 ms, 4 inputs DBBC

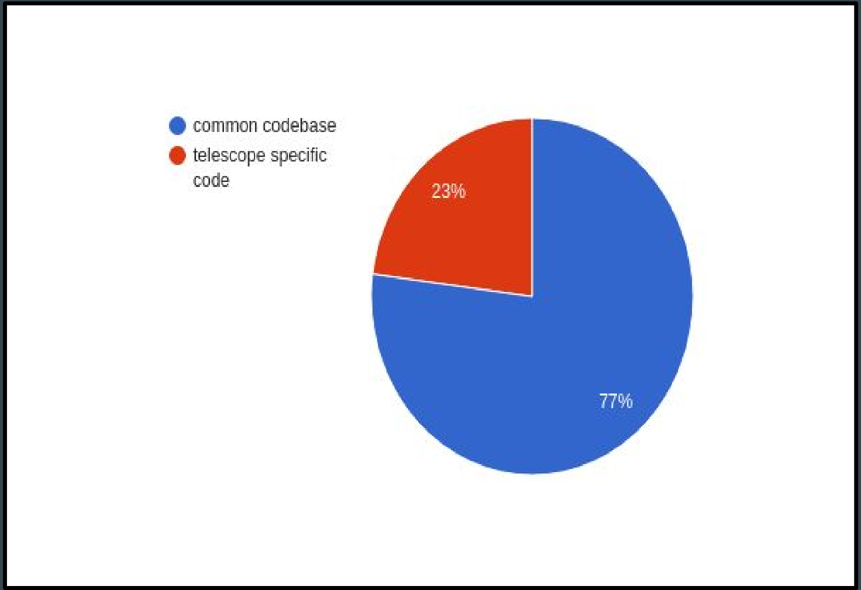
DISCOS features

- Based on ALMA **Common** Software
    - Distributed objects architecture
    - ACS component as the basic unit which performs tasks
    - Components expose interfaces to other components.
- **Common** interfaces design for the three telescopes
- Components organised in subsystems
- Each subsystem has a “boss” component, which has in charge the communications inward and outward the subsystem

DISCOS implementation

- DISCOS - NURAGHE (SRT)
- DISCOS - ESCS (Medicina and Noto)

- A common monolithic codebase (77%):
  - management (scheduling, observing modes)
  - subsystem bosses
  - user interfaces
  - libraries
- Specific code coping differences among telescopes



How big is DISCOS?

Totals grouped by language (dominant language first):

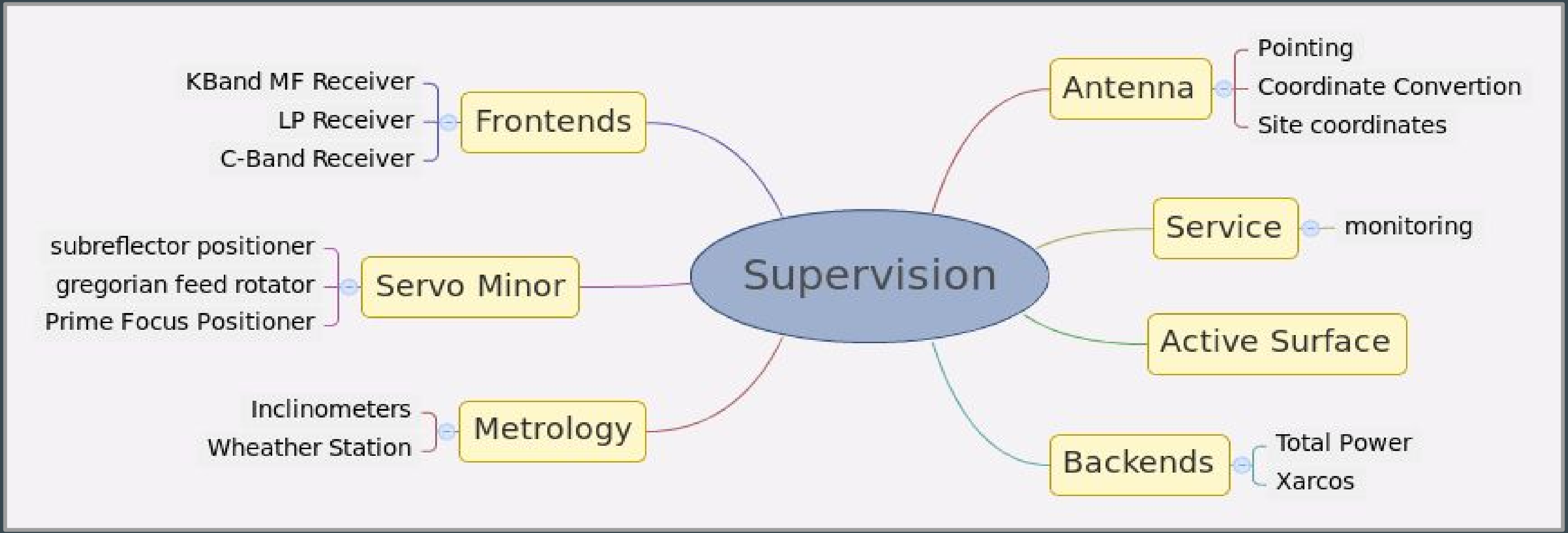
cpp:	383778 (72.59%)
xml:	85988 (16.26%)
ansic:	30854 (5.84%)
python:	26607 (5.03%)
sh:	1328 (0.25%)
fortran:	144 (0.03%)
perl:	14 (0.00%)

Total Physical Source Lines of Code (SLOC) = 528713

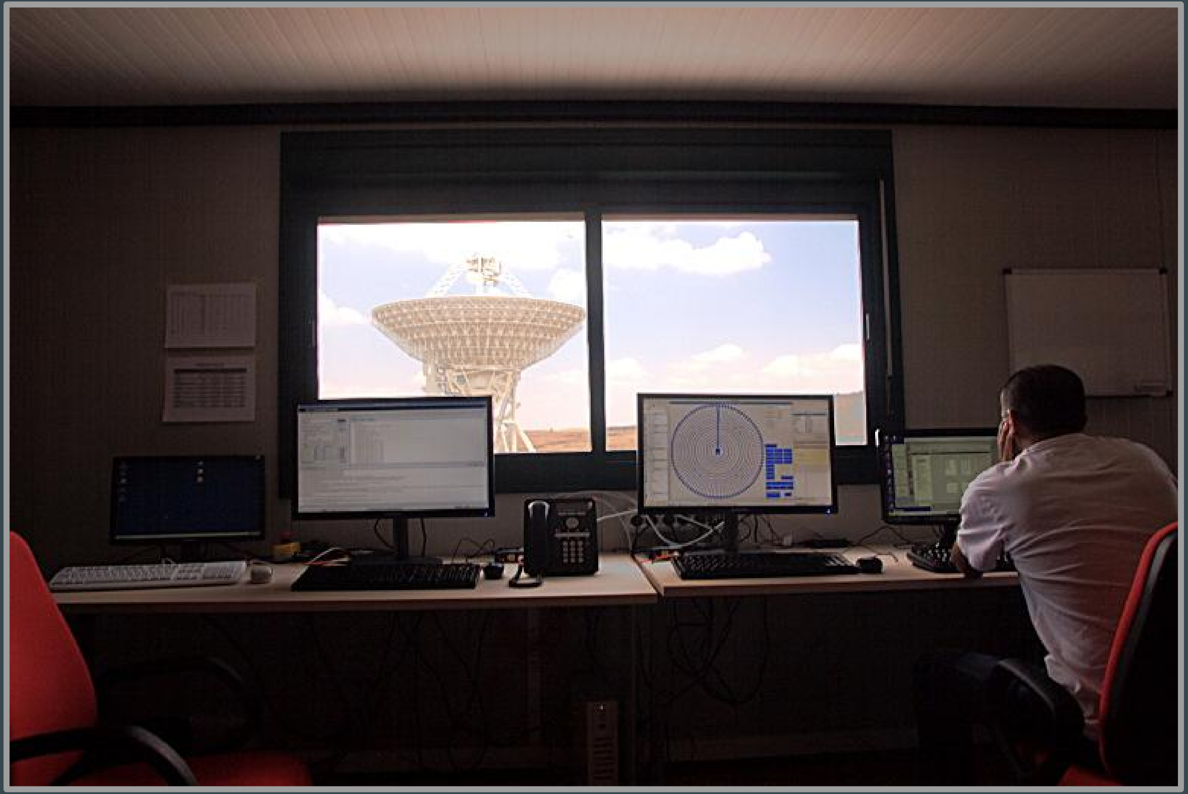
generated using David A. Wheeler's 'SLOCCount'.

statistics by www.openhub.net

DISCOS Design

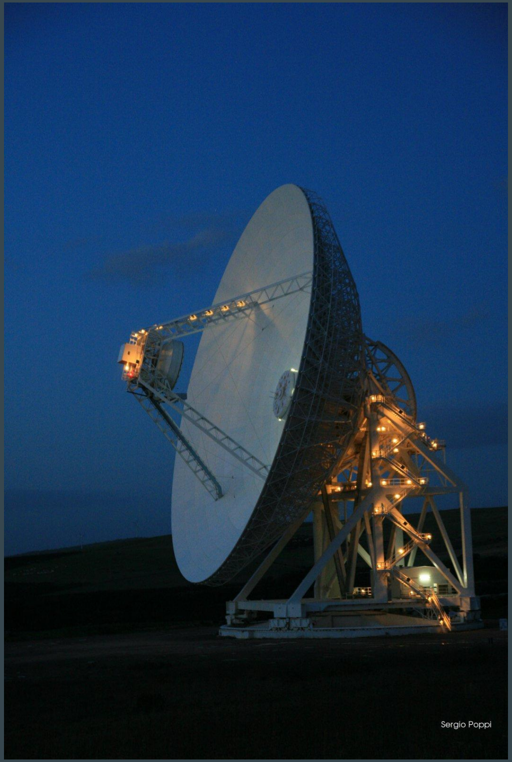


What the control software must do?



It drives the telescope...

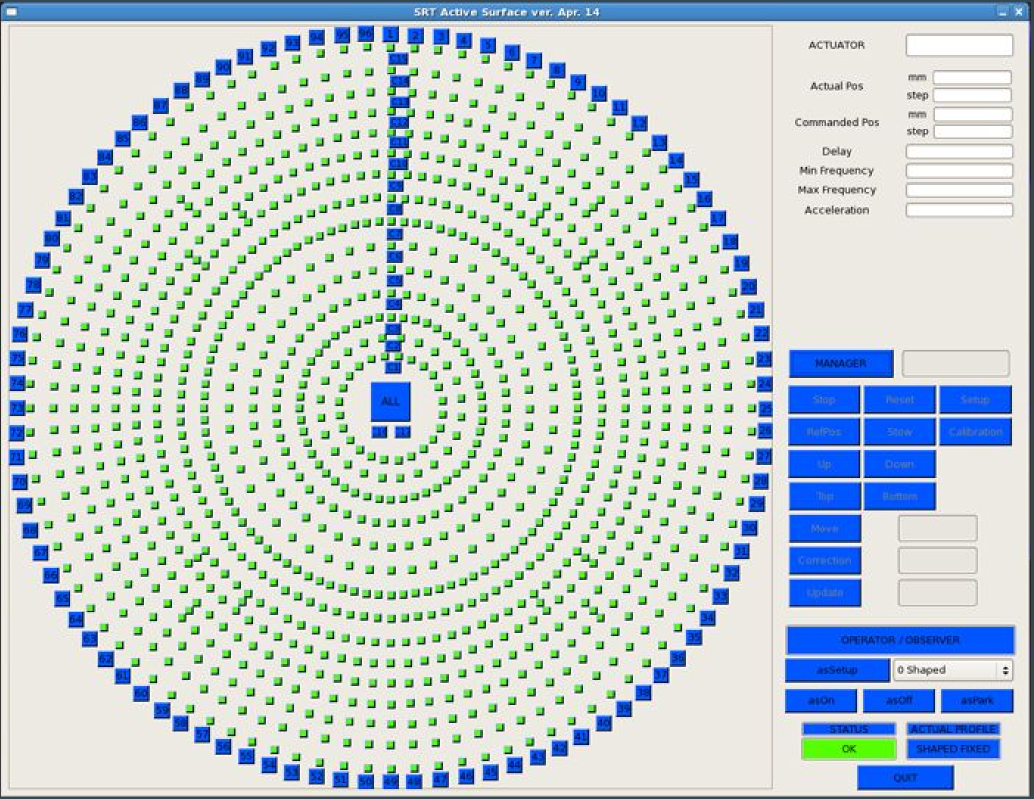
- The antenna control unit needs time tagged azimuth and elevation coordinates
  - The antenna subsystem computes t-tagged coordinates to perform scanning strategies:
    - Sidereal tracking
    - OTF in equatorial, horizontal, and galactic coordinate system



It set the receivers in the right focal position...

- The minor servo subsystem:
- Set the chosen receiver into it focal position
- Drives the subreflector tracking the best focus as the elevation changes

It allows the primary mirror to be in “good shape”

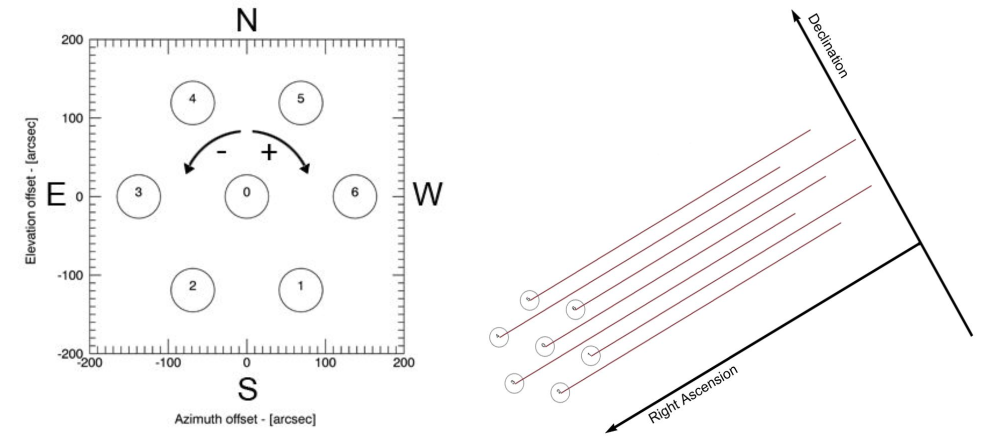


- The AS subsys set each of the 1116 actuators to its elongation for commanded elevation

SRT

It derotates multifeeds

- The Multifeed K-band receiver hosts an hardware derotator to follow the parallactic angle
- If needed, it rotates the receiver following the paralactic angle



Allows data acquisition with housekeeping infos!!!

- Data ara acquired by digital backends (total power, ROACH2 based BE) at high sampling rate (10 ms or faster)
- Housekeeping ancillary infos are mandatory. For each sample there are
  - Data
  - Timestamp
  - Pointing coordinates
  - Weather parameters
  - Goodness of the sample (all the subsystem where working correctly?)

What if the sample rate is one sample every 1 ms?

Discos must continuously

1. Check that the telescope, at time t, is at the right position
2. Check each device, at time t , is at the right position
3. Save the acquired data each 1 ms, together with a status flag (1 if all the check are ok) together with the all the other HK parameters

- Synchronism is critical

Another Critical point: data rate!

- Totalpower BE produces   MAX 64 KB/s
- SARDARA ROACH2 BE (16384) produces 128 MB/s (7-beam mode 900 MB/s)

The data rate is close to the disk rate limit

Optimizing disk and filesystem is mandatory!

Are we facing BIG DATA problems?

Sync it!

Al the subsystem have high accuracy time reference (IRIG-B, 10 MHz, ntp)

ACS has a centralized time system. All the subsystem follow the manager pace

If something fails?

Error backtrace and centralized Logging allow to understand what's happened (or could allow)

Maintenance pitfalls

- Big codebase
- Different product lines:
  - Medicina
  - SRT
  - Noto
- Development and testing during production (SRT)

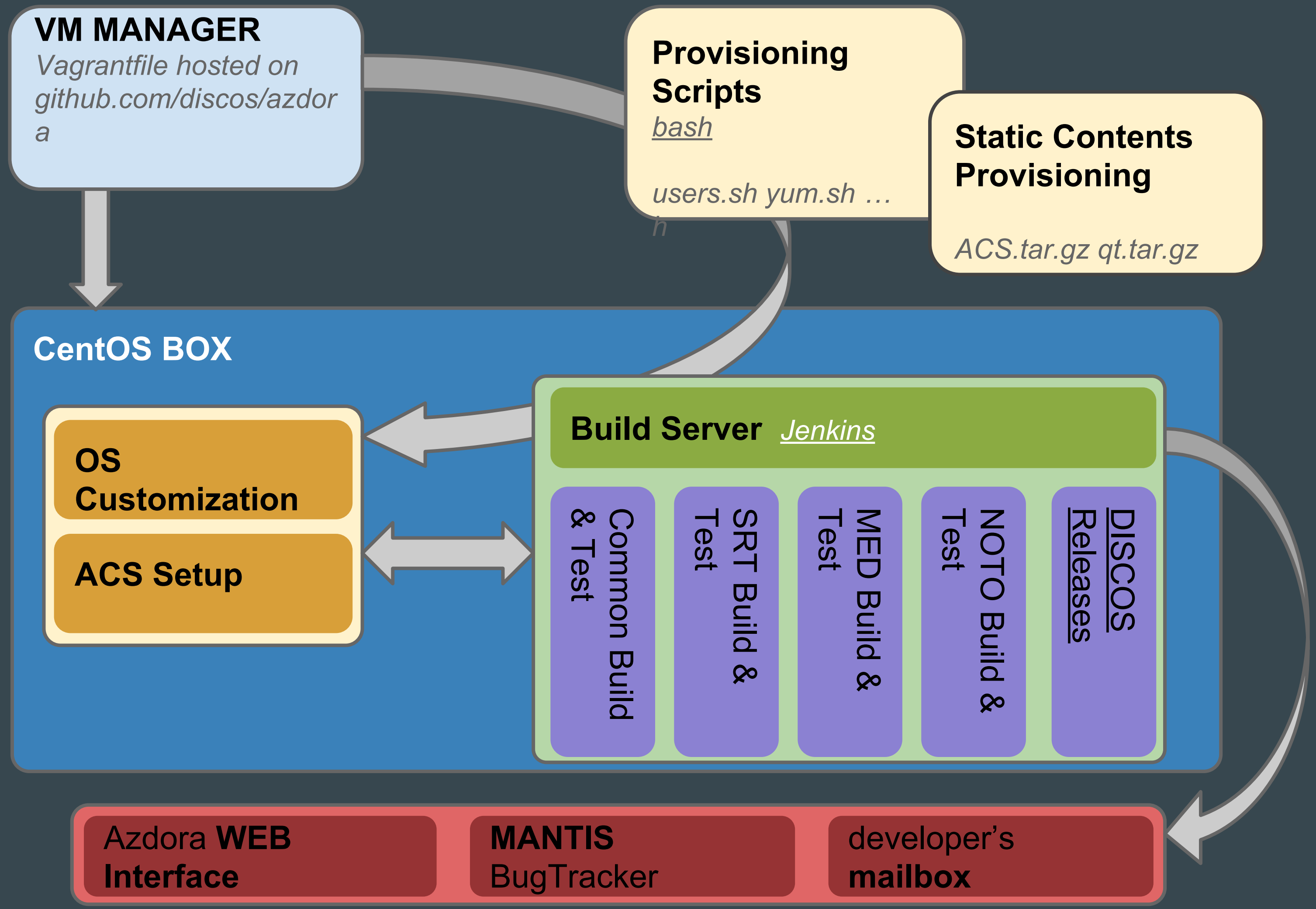
Automatized tasks are needed!

Build the code…

Software repository based on svn

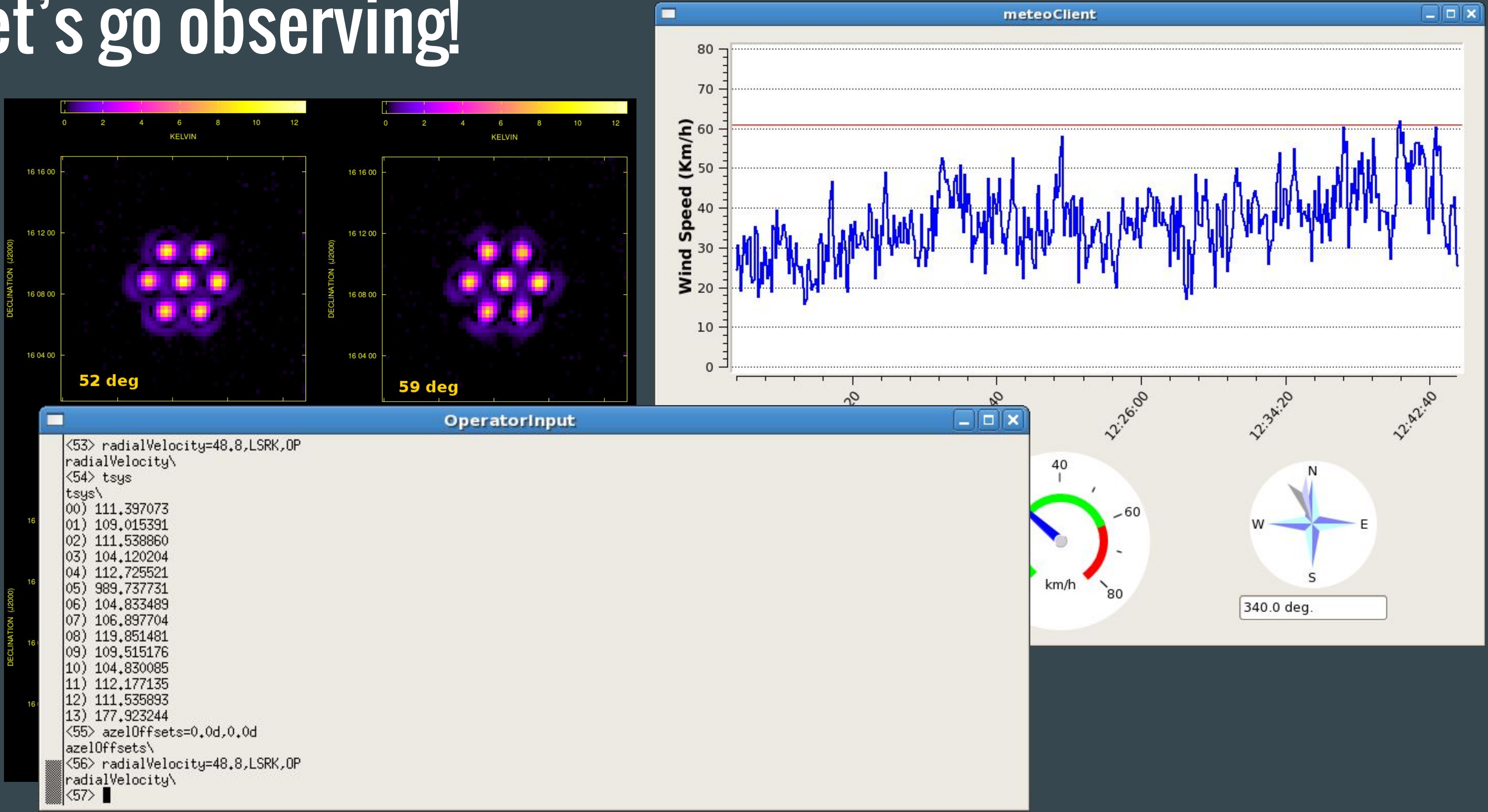
- Build Server AZDORA:
  - Completely automated setup of a virtual machine with ACS installed and configured along with all necessary dependencies
  - Jenkins installation for continuous integration
  - Already configured for accessing nuraghe-devel, mantis BT , github ready
  - Born for testing but can evolve into our standard management platform

Continuous integration



VM MANAGER
Vagrantfile hosted on github.com/discos/azdora
Provisioning Scripts
bash
users.sh yum.sh …
Static Contents Provisioning
ACS.tar.gz qt.tar.gz
CentOS BOX
OS Customization
ACS Setup
Build Server
Jenkins
Common Build & Test
SRT Build & Test
MED Build & Test
NOTO Build & Test
DISCOS Releases
Azdora WEB Interface
MANTIS BugTracker
developer's mailbox

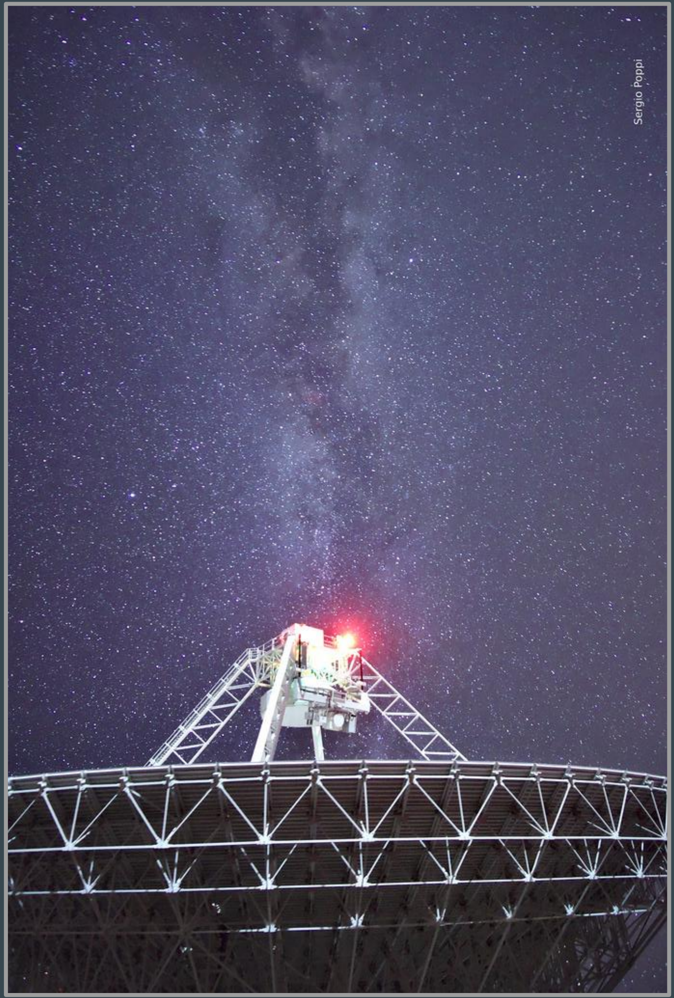
Let's go observing!



Questions?

See:

http://discos.readthedocs.org/



Medicina configuration

- cluster configuration, 4 real nodes, 2 VM nodes
  - 1 - system supervisor
  - 2 - system execution cores
  - 1 - user/project handling
  - 1 - general service (web/VM)
  - 1 - system gui/remote interface
- 1 Gbit private lan dedicated to nodes
- 1 Gbit private lan dedicated to telescope devices
- 10 Gbit lan dedicated to storage and data handling
- user/project management: NIS, NFS
- storage and housekeeping: GPFS -> Lustre
- remote interface: VNC
- virtualization: VirtualBox
- science operativity : IDL, Python

SRT Configuration

- 5 real server
- 1 system supervisor (mng)
- 1 active surface controller (as)
- 1 system gui console (obs1)
- 1 VM (lo)
- 1 user data reduction and QL (local user authentication) obs2
- 1 WS vnc login and data retrieving
- 1 Gbps lan general purpose
- nfs shared disk
- 18 TB mirrored storage
- Svn repository
- Science operativity: IDL, python,