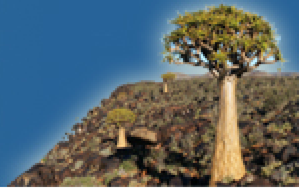
MeerKAT Control-and-Monitoring Design Concepts & Status

Lize van den Heever
MeerKAT CAM: Technical Lead

ICALEPCS 2013
(MOCOAAB06)

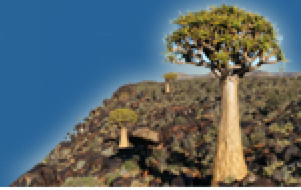
Introducing the MeerKAT Project



- MeerKAT is a 64-dish Radio Telescope:
  - Being built in the Karoo in the Northern Cape of South Africa
  - Is a Precursor for the SKA
  - Has a 7-dish engineering prototype, currently in operation, called KAT-7

- MeerKAT's vision:
  - use Offset Gregorian antennas in a radio telescope array combined with optimized receiver technology in order to achieve superior imaging and maximum sensitivity,
  - be the most sensitive instrument in the world in L-band,
  - be an instrument that will be considered the benchmark for performance and reliability by the scientific community at large, and
  - be a true precursor for the SKA that will be integrated into the SKA-MID dish array.

MeerKAT Project Progress

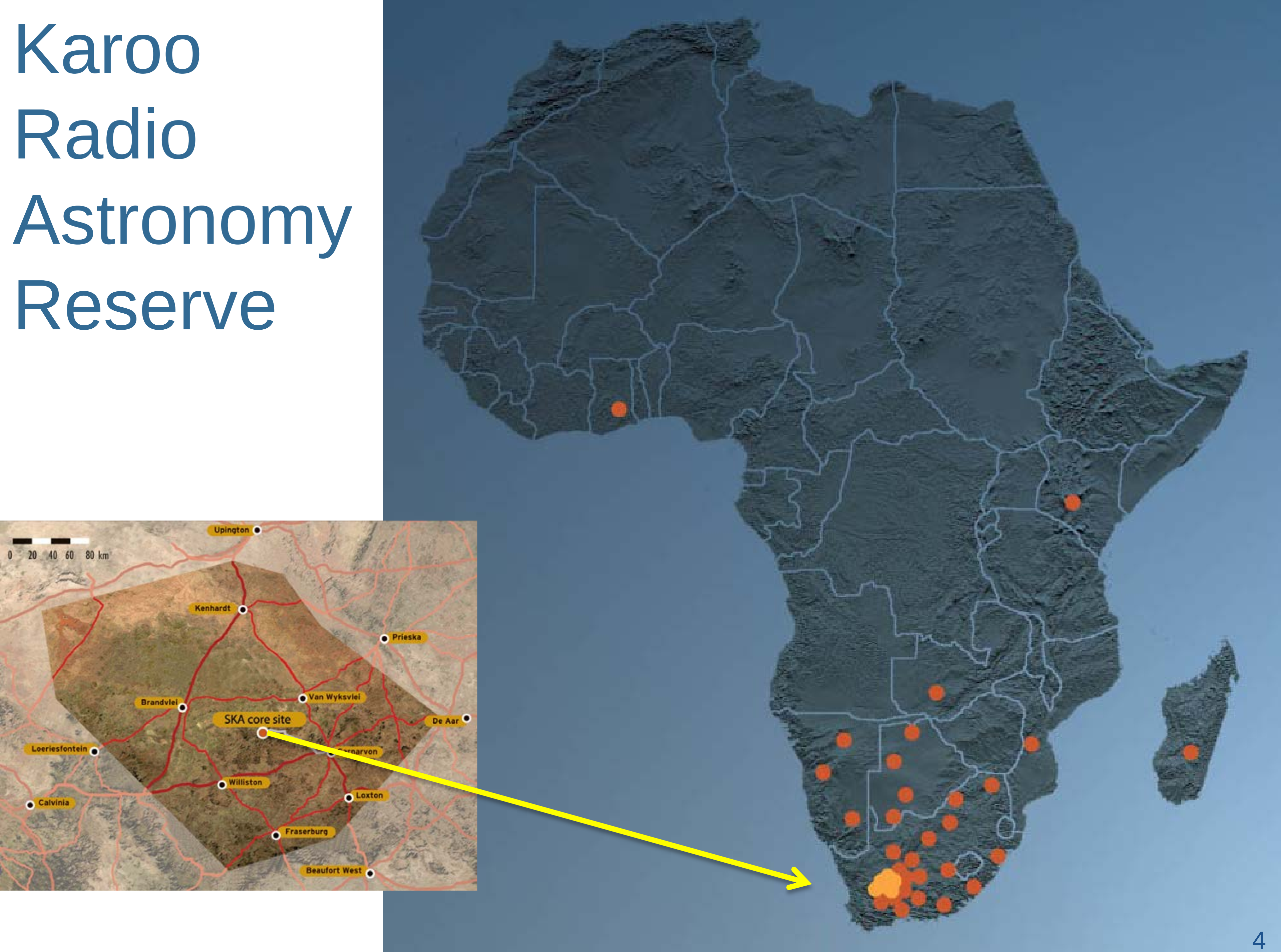


- KAT-7, a 7-dish engineering prototype for MeerKAT
  - is being operated 24x7
  - already producing exciting science and first paper published

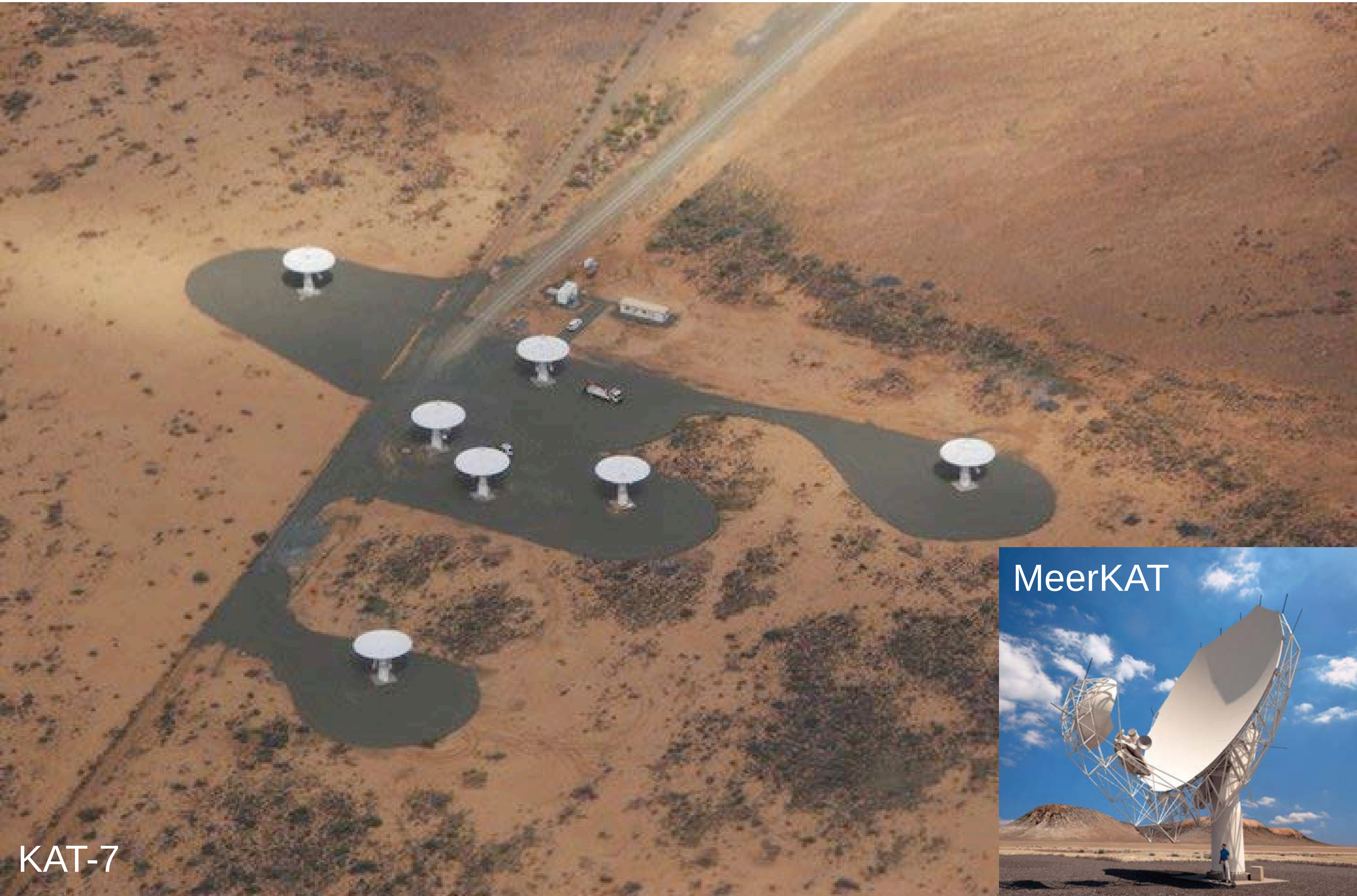
- MeerKAT project
  - commissioning of first MeerKAT antenna will start Mar 2014
  - 4 antennas on site by Dec 2014
  - all 64 antennas installed on site by Dec 2016
  - with 32 antennas fully commissioned by Dec 2016

- MeerKAT CAM (Control-And-Monitoring) subsystem
  - MeerKAT CAM Preliminary Design Review completed in July 2013 with an international panel of domain experts
  - KAT-7 CAM subsystem in place
  - CAM team currently expanding that for the MeerKAT Receptor Test System (first 4 receptors) to be ready by Feb 2014

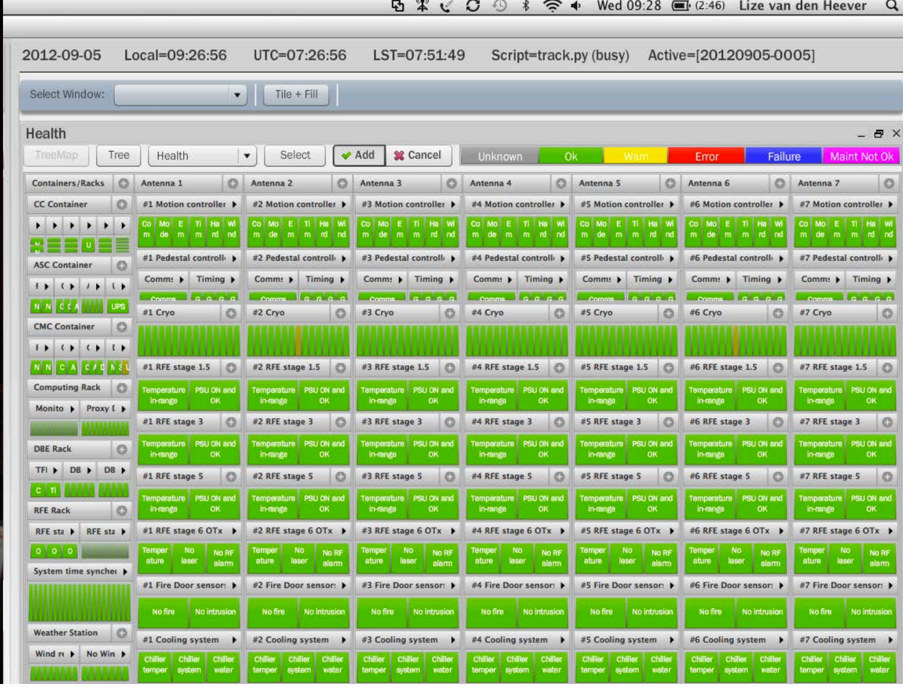
Karoo Radio Astronomy Reserve



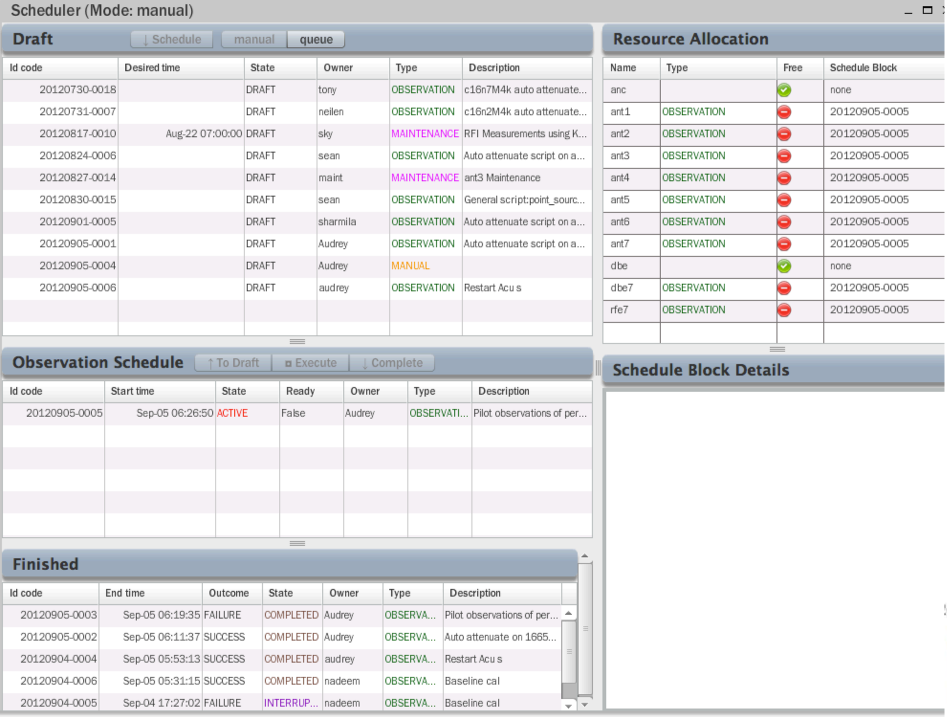
KAT-7 in the Karoo



Operational KAT-7



```
2012-09-05   Local=09:26:56   UTC=07:26:56   LST=07:51:49   Script=track.py (busy)   Active=[20120905-0005]
```



Scheduler (Mode: manual)

Draft | Schedule | manual | queue

Id code	Desired time	State	Owner	Type	Description
20120730-0018		DRAFT	tony	OBSERVATION	c16n7M4k auto attenuate...
20120731-0007		DRAFT	neilen	OBSERVATION	c16n2M4k auto attenuate...
20120817-0010	Aug-22 07:00:00	DRAFT	sky	MAINTENANCE	RFI Measurements using K...
20120824-0006		DRAFT	sean	OBSERVATION	Auto attenuate script on a...
20120827-0014		DRAFT	maint	MAINTENANCE	ant3 Maintenance
20120830-0015		DRAFT	sean	OBSERVATION	General script:point_sourc...
20120901-0005		DRAFT	sharmila	OBSERVATION	Auto attenuate script on a...
20120905-0001		DRAFT	Audrey	OBSERVATION	Auto attenuate script on a...
20120905-0004		DRAFT	Audrey	MANUAL	
20120905-0006		DRAFT	audrey	OBSERVATION	Restart Acu s

Resource Allocation

Name	Type	Free	Schedule Block
anc		✓	none
ant1	OBSERVATION	⊖	20120905-0005
ant2	OBSERVATION	⊖	20120905-0005
ant3	OBSERVATION	⊖	20120905-0005
ant4	OBSERVATION	⊖	20120905-0005
ant5	OBSERVATION	⊖	20120905-0005
ant6	OBSERVATION	⊖	20120905-0005
ant7	OBSERVATION	⊖	20120905-0005
dbe		✓	none
dbe7	OBSERVATION	⊖	20120905-0005
rfe7	OBSERVATION	⊖	20120905-0005

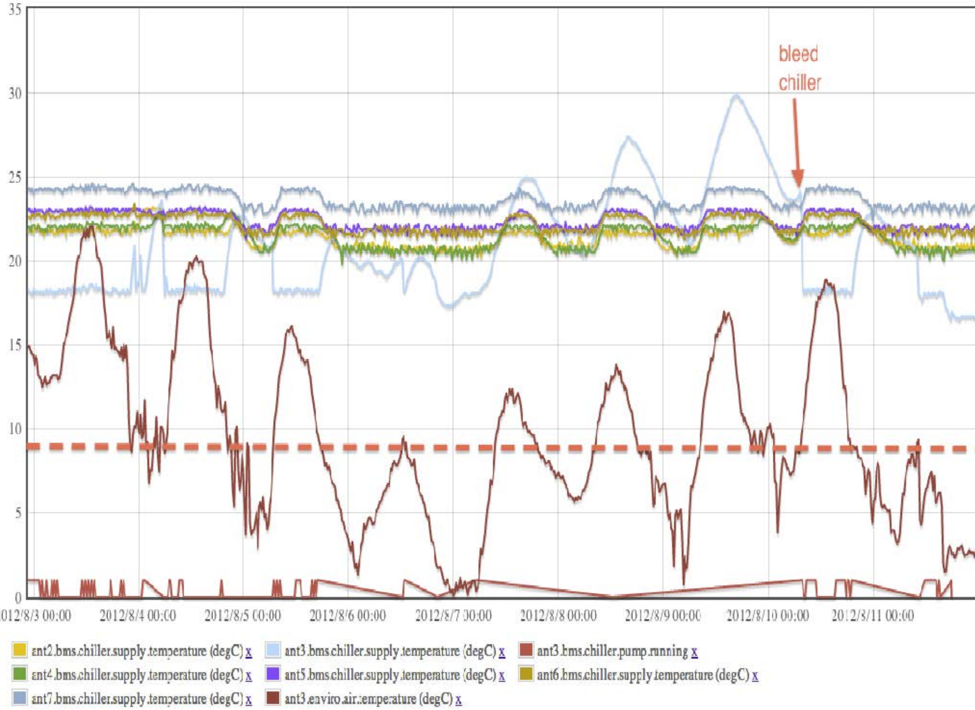
Observation Schedule | To Draft | Execute | Complete

Id code	Start time	State	Ready	Owner	Type	Description
20120905-0005	Sep-05 06:26:50	ACTIVE	False	Audrey	OBSERVATI...	Pilot observations of per...

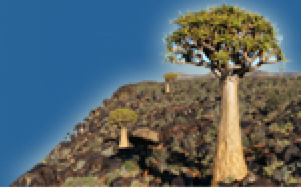
Schedule Block Details

Finished

Id code	End time	Outcome	State	Owner	Type	Description
20120905-0003	Sep-05 06:19:35	FAILURE	COMPLETED	Audrey	OBSERVA...	Pilot observations of per...
20120905-0002	Sep-05 06:11:37	SUCCESS	COMPLETED	Audrey	OBSERVA...	Auto attenuate on 1665...
20120904-0004	Sep-05 05:53:13	SUCCESS	COMPLETED	audrey	OBSERVA...	Restart Acu s
20120904-0006	Sep-05 05:31:15	SUCCESS	COMPLETED	nadeem	OBSERVA...	Baseline cal
20120904-0005	Sep-04 17:27:02	FAILURE	INTERRUP...	nadeem	OBSERVA...	Baseline cal

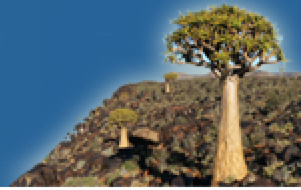


Evolution of MeerKAT CAM



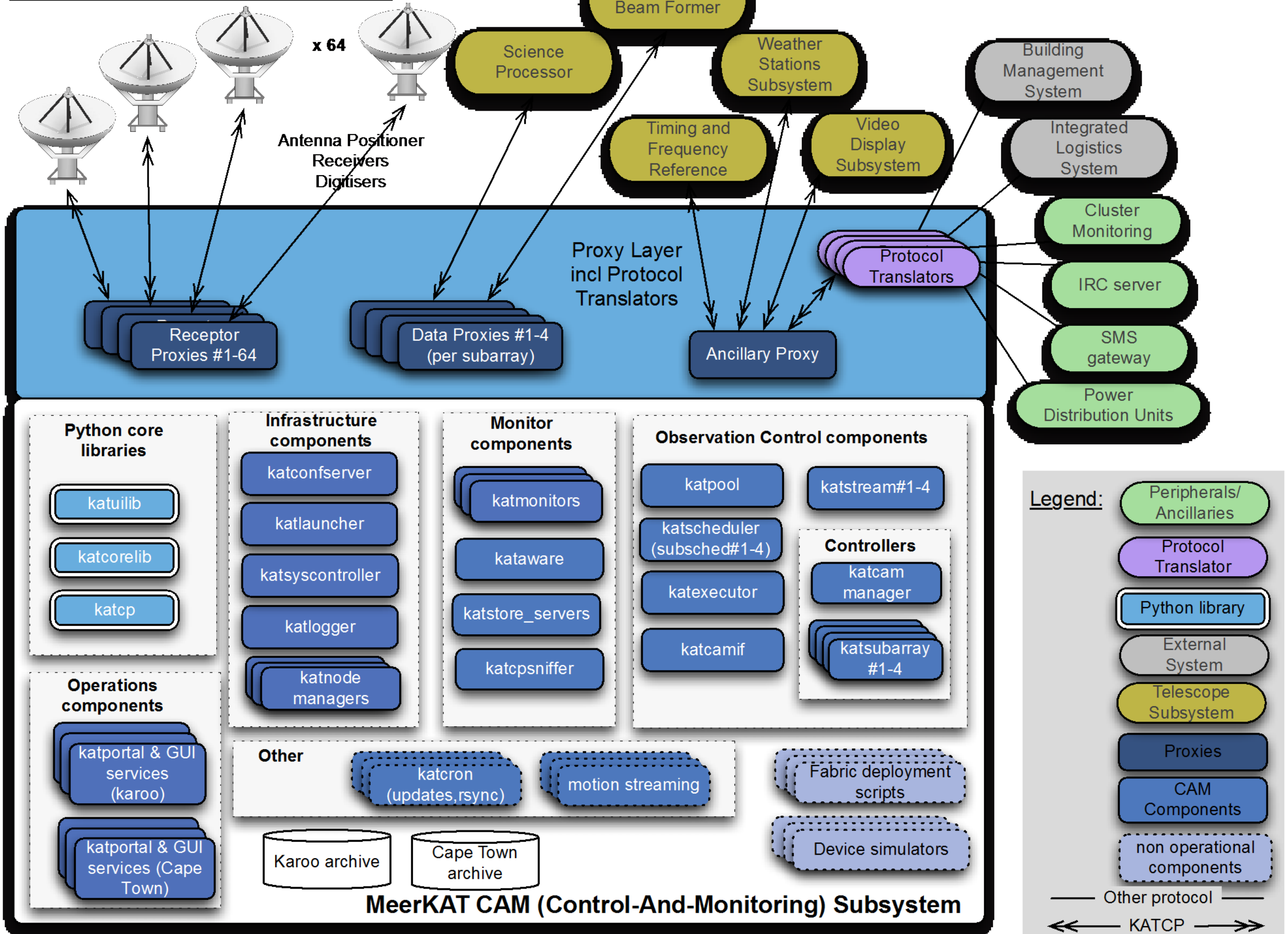
- SKA South Africa:
  - funded by the NRF (National Research Foundation)
  - started in 2004 with an XDM project
  - followed by the Fringe Finder project (the first 2 KAT-7 antennas), completed by end 2009
  - full KAT-7 project followed and is fully operational 24x7 (7 antennas)
  - now busy with the MeerKAT project (64 antennas)
  - has a culture of:
    - learning, improving, enhancing
    - keeping it simple, until proven to be insufficient
    - using open source and creative solutions
- MeerKAT CAM Subsystem
  - Many people involved over the course of these projects
  - Provided ideas for improvements and enhancements of CAM
  - MeerKAT CAM design is a result from all these efforts, not clean sheet
  - Most recently a concerted design effort to fully document and formally review the MeerKAT CAM design
  - Culminated in MeerKAT CAM PDR in July 2014
  - Always view towards scalability to the size of SKA Phase 1 (250 dishes)

MeerKAT Key Design Concepts



1. KATCP for standardised communication
2. Standardised Central logging
3. Proxy Layer and KATCP Device Translators
4. Fully Simulated System
5. Adaptive System based on Interrogation and Discovery
6. Flexible Central System Configuration
7. Homogenous Node Management
8. Soft real-time control with Ethernet as a field bus
9. Hierarchical monitoring and Distributed archiving

MeerKAT CAM Overview



x 64

Antenna Positioner
Receivers
Digitisers

Correlator
Beam Former

Science
Processor

Weather
Stations
Subsystem

Timing and
Frequency
Reference

Video
Display
Subsystem

Building
Management
System

Integrated
Logistics
System

Cluster
Monitoring

IRC server

SMS
gateway

Power
Distribution Units

Proxy Layer
incl Protocol
Translators

Protocol
Translators

Receptor
Proxies #1-64

Data Proxies #1-4
(per subarray)

Ancillary Proxy

Python core libraries

- katuilib
- katcorelib
- katcp

Infrastructure components

- katconfserver
- katlauncher
- katsyscontroller
- katlogger
- katnode managers

Monitor components

- katmonitors
- kataware
- katstore_servers
- katcpsniffer

Observation Control components

- katpool
- katstream#1-4
- katscheduler (subsched#1-4)
- katexecutor
- katcamif

Controllers

- katcam manager
- katsubarray #1-4

Operations components

- katportal & GUI services (karoo)
- katportal & GUI services (Cape Town)

Other

- katcron (updates,rsync)
- motion streaming

Fabric deployment scripts

Device simulators

Karoo archive

Cape Town archive

MeerKAT CAM (Control-And-Monitoring) Subsystem

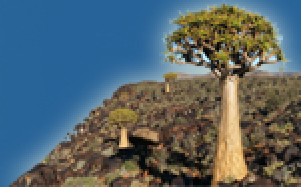
Legend:

- Peripherals/ Ancillaries
- Protocol Translator
- Python library
- External System
- Telescope Subsystem
- Proxies
- CAM Components
- non operational components

—— Other protocol ——

←— KATCP —→

MeerKAT Key Design Concepts



#1. Standardized communications, reporting and logging layer
#2. Discovery of monitoring points and commands on the interface
#3. Adaptive system design adjusts in run-time based on discovery

1. KATCP for standardised communication

- KATCP is a text based, human-readable protocol build on TCP/IP
- Provides discovery of monitoring points/sensors and requests/commands
- Allows different sensor strategies (sampling rates) per client, supporting different users to configure different update rates
- Sensor update is a timestamp, status, value combination
- Includes standardised logging and failure reporting
- Publicly release on PyPi

- KATCP is specified as the CAM interface for all subcontracted and internal hardware devices and subsystems
- Also used for internal communication between CAM components

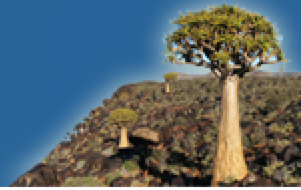
MeerKAT Key Design Concept #2

2. Standardised Central logging

- KATCP guidelines specifies standardized logging and failure logging for devices/subsystems
- This includes logging levels, logging format, type of information expected at each level
- CAM proxy layer exposes device logs for central logging
- All logs from proxies and CAM components are stored centrally
- Level of KATCP logging for each device is configurable via KATCP interface
- Ensures:
  * a consistent mechanism and formatting for system-wide logs
  * a central store of system logs to support fault finding and engineering tests

- CAM provides a web interface for viewing on-line system logs, filtered by source and log levels by the user.

MeerKAT Key Design Concept #3



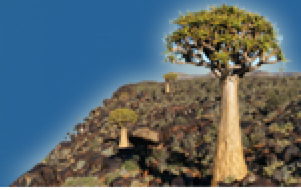
3. Proxy Layer and KATCP Device Translators

- Protects hardware devices and MeerKAT subsystems from direct access
- All engineering/support/system components/tools connect via the proxy layer and not directly to hardware devices/subsystems.
- Proxy may implement special configuration/control for a device (e.g. the Receptor proxy implements pointing corrections for antenna pointing, and the Data proxy implements delay calculations and gain corrections for the Correlator).
- Proxy layer also gathers the KATCP logs from devices for central logging
- Proxy layer provides rolled-up reporting across all devices it manage

Device Translators:

- MeerKAT specified KATCP interface and KATCP simulators for all subcontractors and subsystems
- Device Translators convert specific protocols (like modbus, OPC, web-services, Ganglia metrics) to KATCP, where required
- Allows for the CAM team to develop against a fully Simulated system
- CAM system can be functionally exercised in a fully simulated environment
- Used for CAM functional qualification and operator training

MeerKAT Key Design Concept #4



4. Adaptive System based on Interrogation/Discovery

- KATCP supports discovery of sensors and commands, down to device level
- By design CAM exploits this in-time discovery on all levels and extends that by adapting to the discovered interface in real-time.
- Newly discovered sensors are automatically included throughout the CAM system (without a single change in configuration or lines of code):
  - * sampled and added to archiving
  - * included in rolled-up reporting, including generic alarms
  - * automatically available when plotting updates or extracting history
  - * added to health and status displays & views through rolled-up sensors
- Newly discovered commands are automatically included in CAM low level device control, available to engineers and expert users immediately
- Even adding a new CAM component to be monitored needs nothing more than defining the component in the configuration
- Adding a new simple device to a proxy needs nothing more than defining the device name and location in the configuration
- This adaptive design based on discovery allows for seamless integration as new versions of controllers/hardware are rolled out

MeerKAT Key Design Concept #5 & #6

5. Fully Simulated System

- Fully simulated system up to the KATCP interface of each device & subsystem
- Allows full software development without dependency on any hardware
- Simulators implement full KATCP interface and representative behaviour

6. Flexible Central System Configuration

- Powerful and flexible system configuration in human readable text files
- Supports integration and incremental rollout of receptors
- Can run any combination of real and simulated devices as CAM “sites”
- Includes identification of servers and virtual notes participating in the “site”
- Includes configurable health displays, aggregate sensors with user defined programmatic rules, sampling strategies for monitoring and archiving and alarm configurations and actions.

MeerKAT Key Design Concept #7

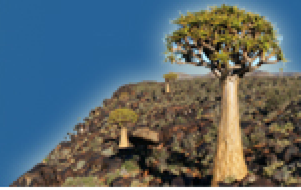
7. Homogenous Node Management

- CAM implements homogeneous node management across all nodes (VMs)
- A single **headnode** acts as system controller
- Headnode coordinates the system from central configuration
- Same suite of software is deployed on all nodes
- Each node (including headnode) starts up with only a **katnodemanager** service
- Each katnodemanager waits for headnode to register the subset of CAM processes to run on that node and for launch instructions

Allows for seamless scaling of servers when performance demands it

- Extremely easy to add new servers that host more virtual nodes
- Only need to update the central configuration to identify new servers with new virutal nodes and distribute the processes to run on each virtual node
- Then restart, no code changes required

MeerKAT Key Design Concept #8



8. Soft real-time control with Ethernet as a field bus

- implies that there are no tight critical control loops in the MeerKAT CAM
- where necessary, real-time control is decentralized to devices
- CAM subsystem issues commands to devices with a specific timestamp

MeerKAT Key Design Concept #9

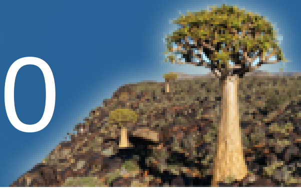
9. Hierarchical monitoring

- Based on standardisation and commonality defined in KATCP guidelines
- Includes standardised failure reporting and failure logging, logging,
- Includes standardised device status & health reporting, and rolled-up reporting on device level
- Each proxy implements rolled up reporting across devices they manage
- Each node manager implements rolled-up reporting for all CAM processes it manages

Standardisation and consistency simplifies the CAM design:

- Provide single points of monitoring to roll-up in hierarchical health reporting, which can be discovered through naming conventions
- Drill down only required for fault finding or when interested in lower level information
- Rolled-up reporting, allows high-level monitoring with hierarchical drill down when required.

MeerKAT Key Design Concept #10



10. Distributed archiving

- A local **katmonitor** component on each virtual CAM node
- Gathers and archives the sensors of all components running on that node
- Each katmonitor writes its sensor updates to a central **katstore** archive through network file system mounts routed through the bulk network, avoiding network traffic bottlenecks
- New nodes are automatically included in the system monitoring and archiving by simply adding the node to the configuration and running an instance of **katmonitor** on that node.

Questions?

lvdheever@ska.ac