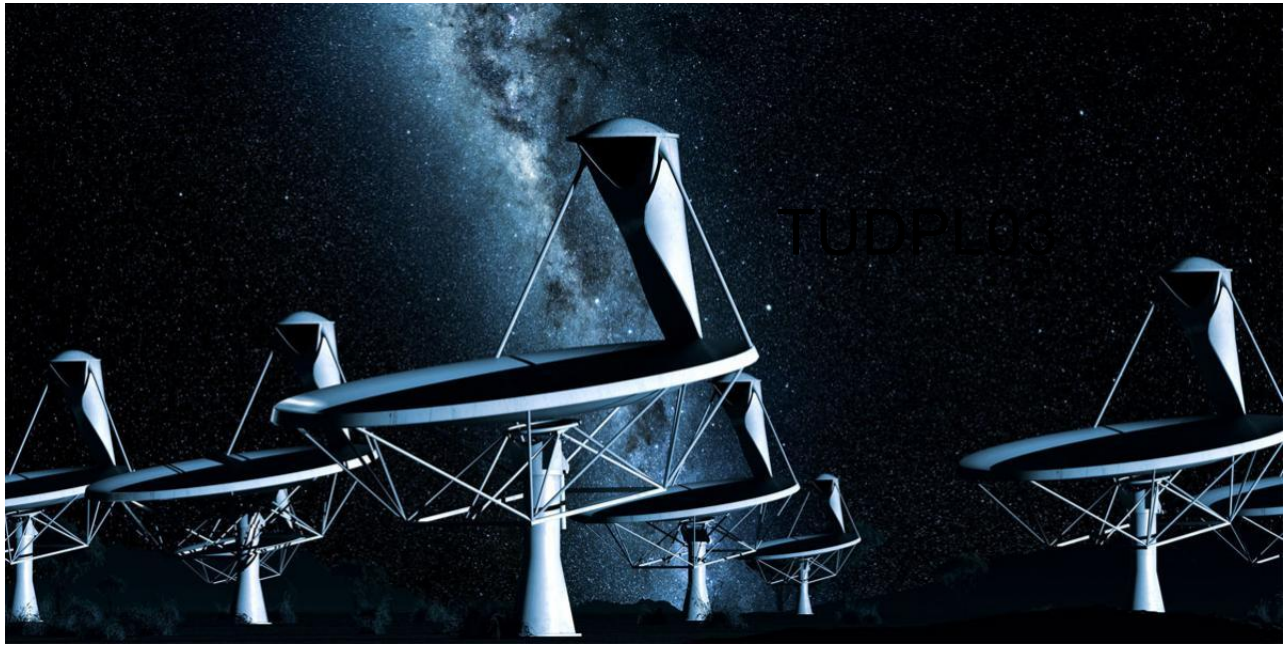
# LMC Simulation Framework (TUDPL03)




ICALEPCS2017
Barcelona · Spain, October 8-13
Palau de Congressos de Catalunya

*This work is based on the research supported by the National Research Foundation (through a flagship project of the SA-INDO collaboration). Any opinion, finding and conclusion or recommendation expressed in this material is that of the author(s) and the NRF does not accept any liability in this regard*

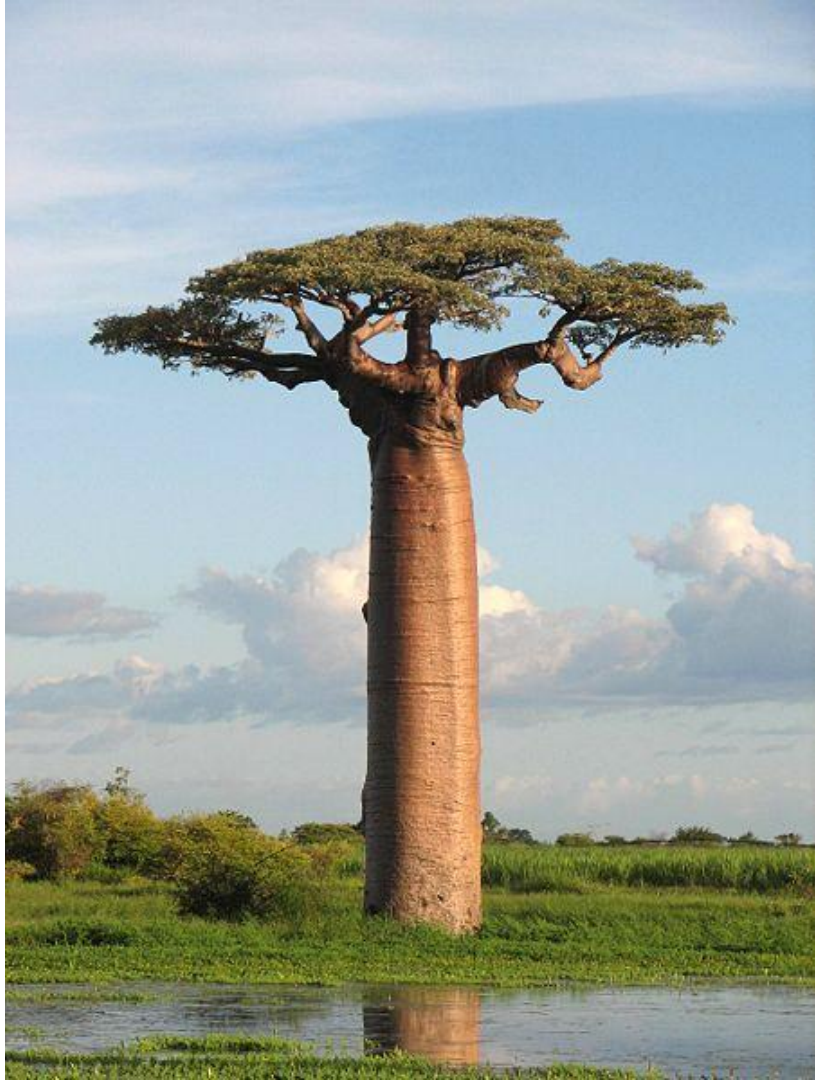
Athanaseus Ramaila
Lize van den Heever
Katleho Madisa
Neilen Marais

# Outline

- Iteration 1
  - *MeerKAT CAM using Tango simulators and protocol translators*
- Iteration 2
  - *Evaluate TANGO tool capabilities in the context of a MeerKAT-like radio telescope*
- Iteration 3
  - *Improved Data-driven Simulation framework, including behaviour extension*
- Iteration 4
  - *Integrating SKA DSH into MeerKAT using protocol translators*

# ITERATION 1

*MeerKAT CAM using Tango simulators and protocol translators*

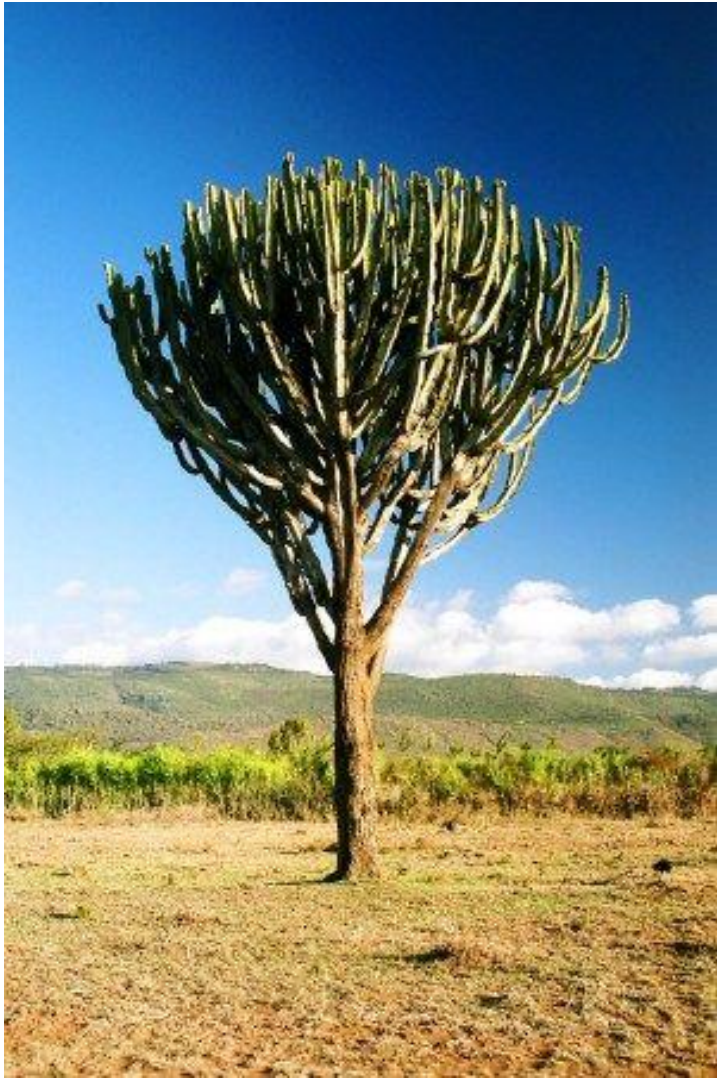


# Iteration 1 Learnings

- Generic TANGO -> KATCP translator works well
  - KATCP "device model" mostly a superset of TANGO "device model"
- Easy to integrate TANGO devices in existing MeerKAT control-and-monitoring system
- Allows for early SKA subsystems testing with MeerKAT, e.g. integrating the first SKA dish and testing it in MeerKAT

# ITERATION 2

*Evaluate TANGO tool capabilities*
*in the context of a MeerKAT-like radio telescope*



# Iteration 2 Learnings

- TANGO community ecosystem provides useful tools
  - TANGO framework architecturally very similar to MeerKAT CAM architecture
- Generic KATCP -> TANGO translator works OK
  - Some KATCP "device model" features were hard to represent in TANGO
  - Could potentially be addressed (pipes?)
- Potential use for SKA <-> KATCP interop

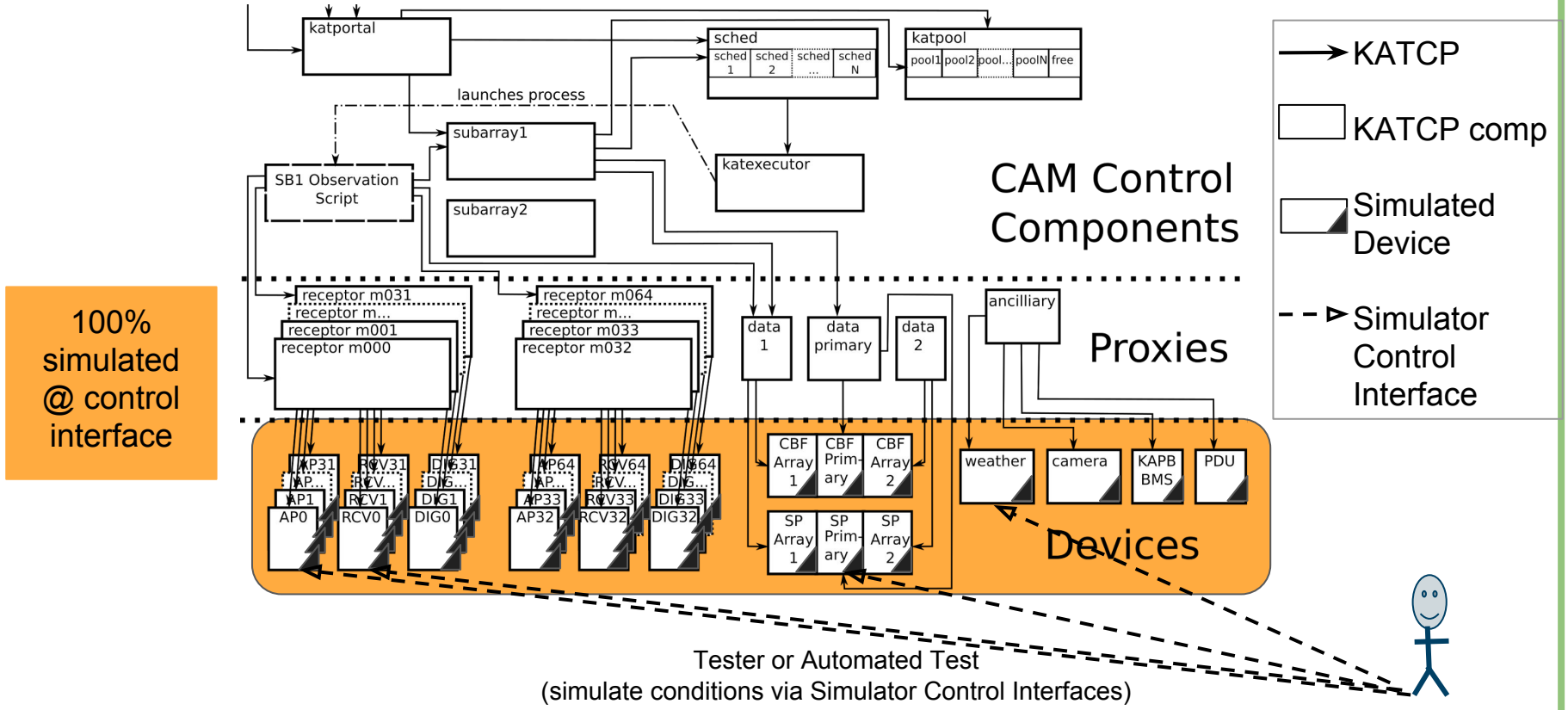
# ITERATION 3

*Improved Data-driven Simulator framework, including behaviour extension*

# Why Simulated Devices

- Positive MeerKAT / KAT-7 experience
- Can be used by the SKA Element Consortia to develop LMC simulators
- Support early development work of Element LMCs (Local Monitor&Control)
- Also used in the SKA Telescope Manager (TM) test environment
- Enable SKA TM  to support early Assembly, Integration and Verification efforts
- Easily configure fully simulated development environments
  - SKA Telescope Manager Development
  - Automated functional/ integration testing
  - Lab integration with partial simulation

# Development / simulated MeerKAT Architecture



# How the Simulator is launched

## Step 0

$ pip install tango-simlib

## Step 1

Basic Sim Spec (POGO xmi)
Complex Sim Spec (SimDD)

*(xmi specify device API,*
*SimDD specify simulator behaviour)*

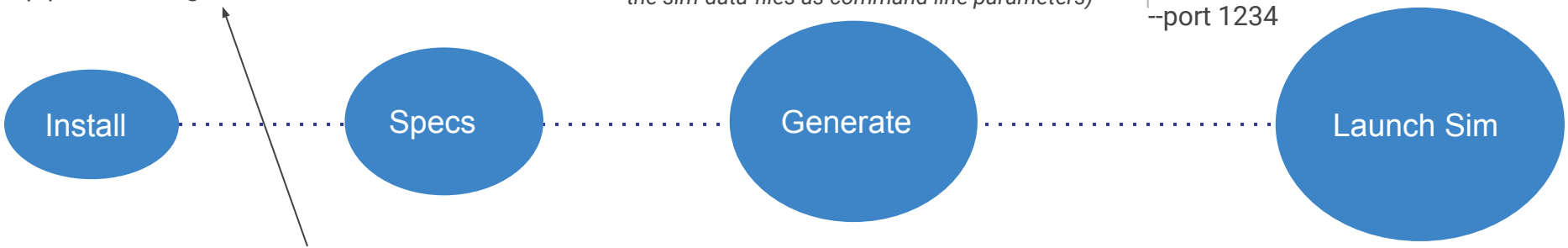
## Step 2

```
$ tango-simlib-simulator-generator \
--sim-data-file Dish.xmi \
--sim-data-file Dish_SIMDD.json \
--device-name DishElement-DS \
--directory /usr/local/bin/
```

*(to generate the TANGO device server script taking the sim-data-files as command line parameters)*

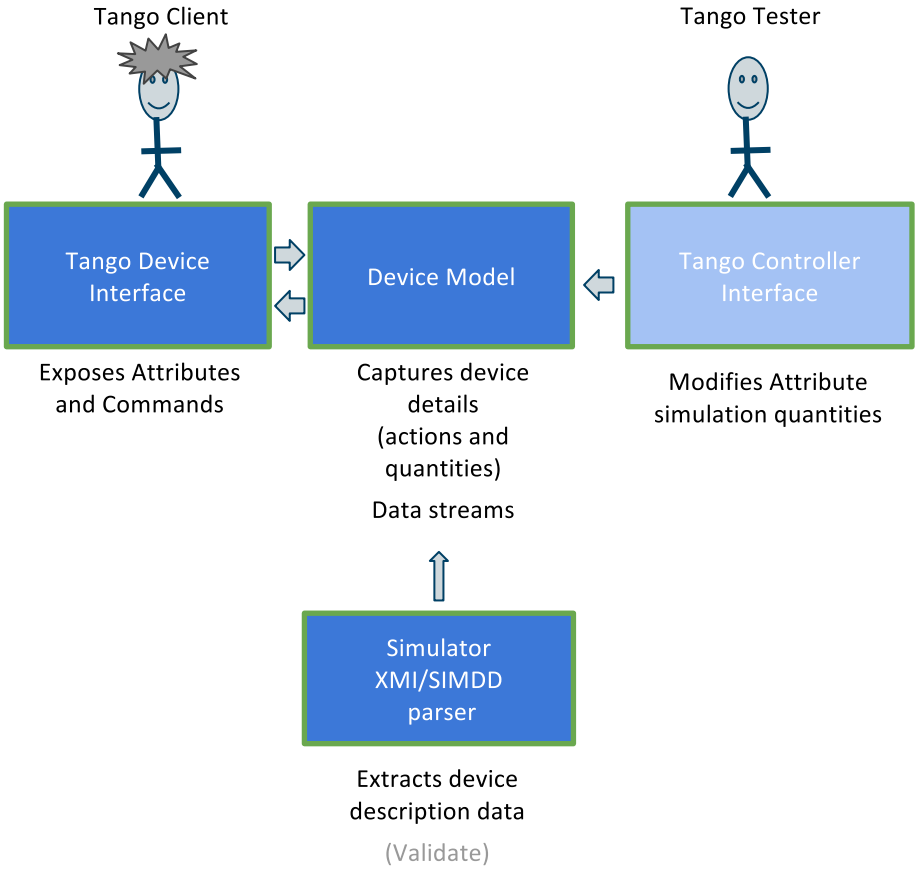
## Step 3

```
# If already  registered in tango db
$ DishElement-DS  dish-000

# auto-register
$ tango-simlib-launcher  \
--name DSH/element/ap000 \
--class DishElement \          --name
DSH/element_ctrl/ap000 \ --class
DishElementControl \
--server-command  DishElement-DS\
--server-instance dish-000 \
--port 1234
```




https://github.com/ska-sa/tango-simlib

# Simulator with a Simulator Controller Interface



# Basic Simulators

- Uses only the Tango POGO interface generation tool (XMI file)
- Attributes are mapped to model quantities without writing any code
- Commands are mapped to default no-op model actions
- Simulation control interface included, used to manipulate the simulator and induce conditions/failures

# Complex Simulators

- Uses simulated device description format to describe simulator behaviour (SimDD JSON file)
- Specify attribute parameters and quantity simulation types
- Override or Modify default command actions using the SimDD
- Custom action handler overrides can be coded in Python
- Simulation control interface, as above

# Simulation Parameters

```
"dynamicAttributes": [
        {
            "basicAttributeData": {
                "name": "an_attribute",
                // Other “standard” attribute parameters that can also
                // be specified in the XMI file has been removed for brevity
                "dataSimulationParameters": {
              "gaussianSlewLimited": {
                 "min_bound": "minimum bound for randomly varied simulator quantity",
                 "max_bound": "maximum bound for randomly varied simulator quantity",
                 "mean": "mean value",
                 "max_slew_rate": "maximum slew rate",
                 "update_period": "update period"
                    }
                },
    ]
```

*Basic attribute simulation categories:*

- *guassianSlewLimited - min/max bounds, mean value, slew_rate and update_period*
- *constantQuantity - initial_value, attribute_quality*

*Basic command simulation categories:*

- *Input parameter transform - Take an input parameter, applies a transform and place output in a temporary variable*
- *Side effect - Simple action that can modify a simulation quantity or internal state variable*
- *Output return - Return value or exception*

# Complex Simulators

*To simulate more complex behaviour the commands can be overridden by implementing and specifying an Override Class.*

*This allows for full flexibility as the complete simulation model can be replaced, if required.*

*Override actions in the override class are prefixed with action then the name of the command on the TANGO device.*

Simulator Data-Description file (SIMDD.json)

```
"override_class": {
    "name": "unique_override_identifier",
    // "module_directory": "Locate the override module in this directory [optional]",
    "module_name": "tango_simlib.examples.override_class"
    "class_name": "OverrideDish"
}
```

Override class

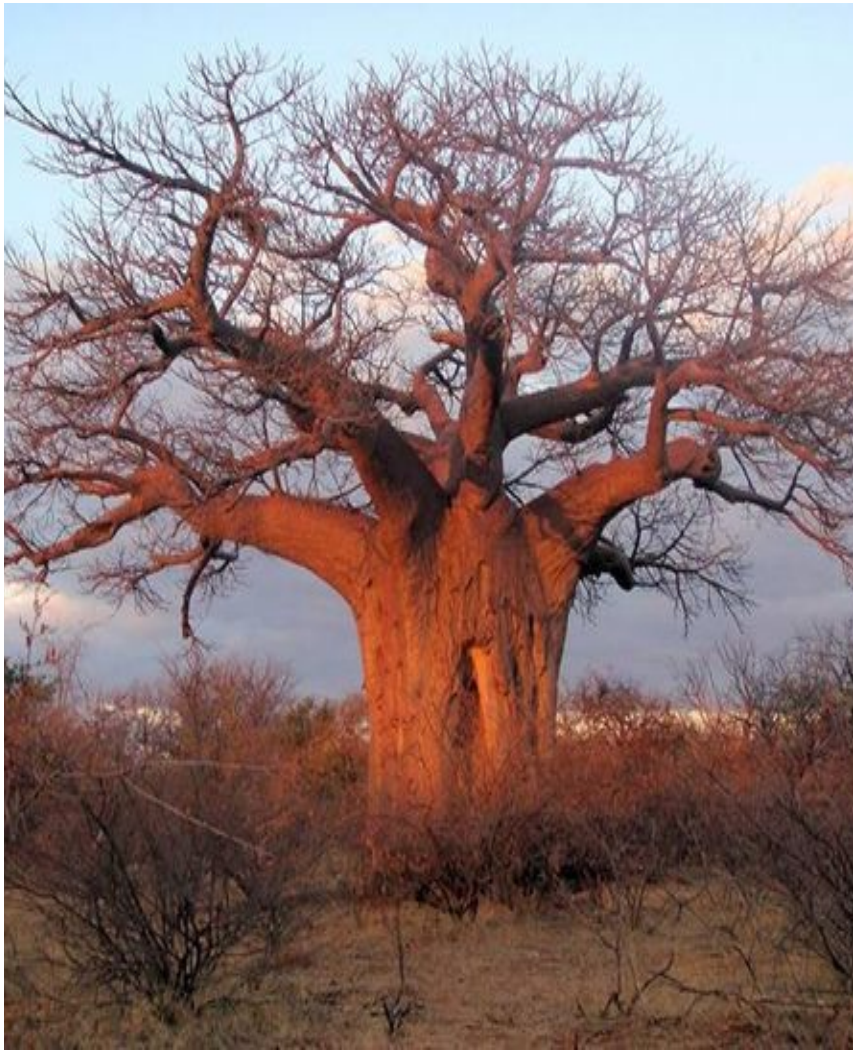
```
class OverrideDish(object):
    """An example of the override class for the TANGO device class 'SkaDishMaster'.
    It provides implementations of the command handler functions for the commands
    specified in the POGO generated XMI data description file.
    """
    def action_slew(self, model, tango_dev=None, data_input=None):
        """The Dish is tracking the commanded pointing positions within the specified
        TRACK pointing accuracy.


        data_input: list
           [Timestamp]
           [azimuth]
           [elevation]
        """
        _allowed_modes = ('OPERATE')
        :
```

# Iteration 3 Learnings

- Ported MeerKAT simulator+test interface model to TANGO
  - Released as FOSS : https://github.com/ska-sa/tango-simlib
- Simple TANGO device simulators are easy to generate - with API from POGO XMI files
  - MeerKAT experience: covers 80% of use cases
- Complex simulators can leverage base functionality - additional behaviour as per SIMDD.json spec

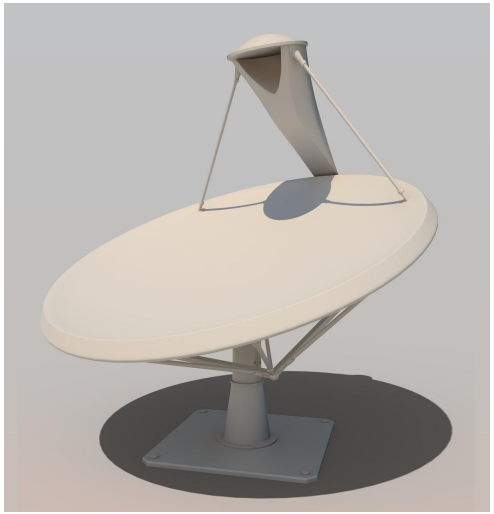
# ITERATION 4

*Integrating SKA DSH into MeerKAT using protocol translators*

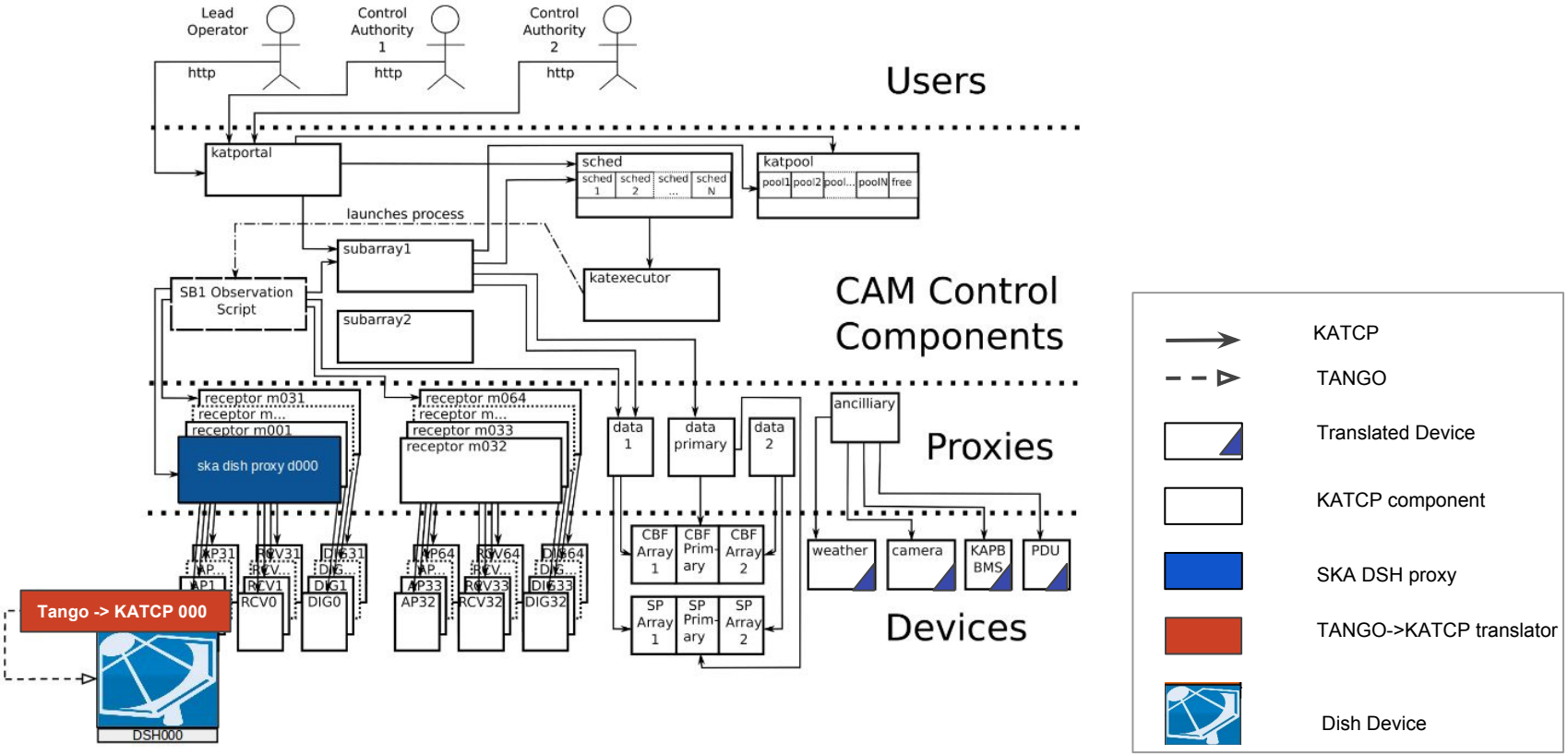


# Integrate SKA DSH Element simulator in MeerKAT

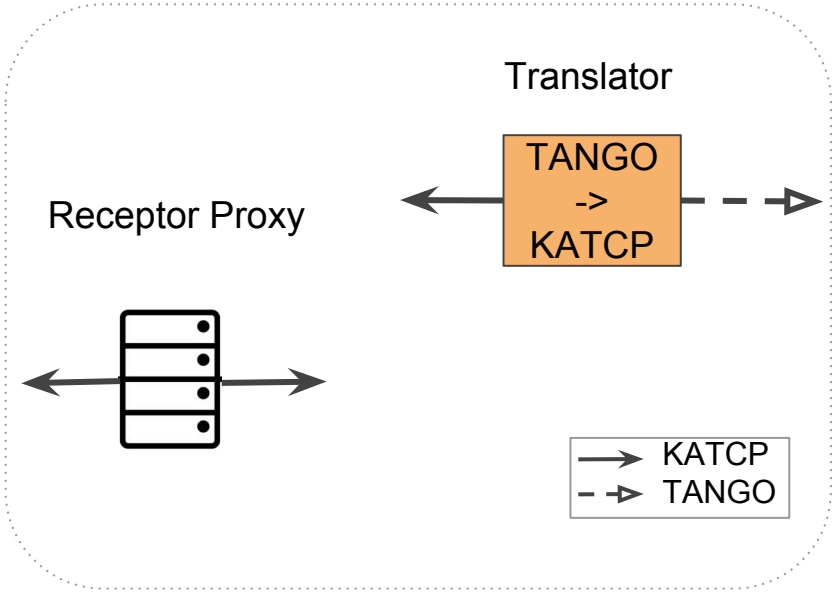
- Will be first (real) element to be integrated
- Prototype to be qualified with MeerKAT
- Preliminary DSH ICD available
- Ported the MeerKAT Antenna Positioner physical model to tango-simlib based DSH simulator
- To prepare MeerKAT for DSH prototype:
  - Integrate DSH simulator with MeerKAT using TANGO->KATCP translator
  - Update MeerKAT DSH proxy for specific DSH behaviour where it differs from MeerKAT receptor behaviour



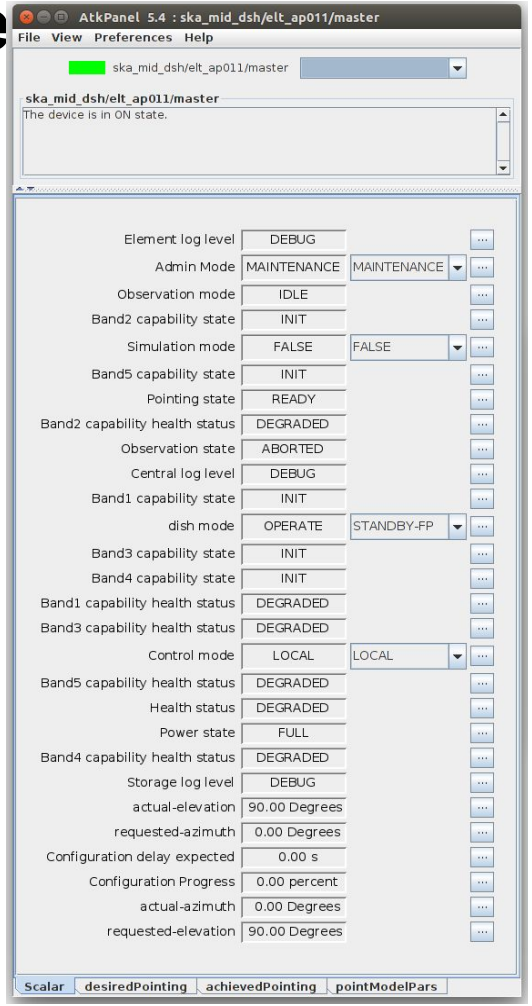
# MeerKAT CAM system with TANGO integration



# Connecting To Dish Simulator Device

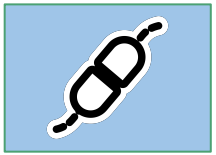




Dish Simulator



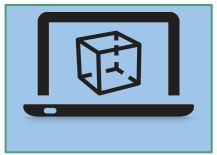
# SKA DISH Proxy

- Provides standardised MeerKAT/KAT-7 high-level interface
  - Development based on existing MeerKAT receptor proxy



System component/tools (TM) connect via the receptor proxy (LMC) not directly to DISH

Allows the rest of the MeerKAT CAM system to interface with the DISH device.



The proxy is responsible for managing devices and exposing their KATCP interface.

Allows simultaneous observation with MeerKAT receptors and prototype DISH

# Data-driven simulator tools


- **DSEE**: Generates Simulator Description files (SimDD)
  - https://gitlab.com/patwari.puneet.ska/MAC-SEEN
- **tango-simlib**: Simulator interface as per SimDD
  and Controller interface to manipulate the simulator
  - https://github.com/ska-sa/tango-simlib
- **mkat-tango**: TANGO/KATCP Device Translators

See ICALEPCS PAPER **TUDPL03** + POSTER **THSH201**


Thank you!