science & innovation
Department:
Science and Innovation
REPUBLIC OF SOUTH AFRICA

National Research Foundation

SARAO
South African Radio Astronomy Observatory

SOFTWARE TESTING ON AN ASTRONOMICAL SCALE

September 2022 / CASPER 2022

James Smith

www.sarao.ac.za

The South African Radio Astronomy Observatory (SARAO) is a National Facility managed by the National Research Foundation and incorporates all national radio astronomy telescopes and programmes.

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

You’re familiar with...

- Software Testing

You're familiar with...

- Software Testing
- Reasons and benefits

You’re familiar with...

- Software Testing
- Reasons and benefits
- Tools

You’re familiar with...

- Software Testing
- Reasons and benefits
- Tools
  - pytest

You’re familiar with...

- Software Testing
- Reasons and benefits
- Tools
    - pytest
    - but there are others!

Main Takeaways

- Testing is a great concept.

Main Takeaways

- Testing is a great concept.
- Methodology and tools can be applied to much bigger systems than just code snippets.

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

Re-cap

- MeerKAT Radio Telescope is getting an extension.

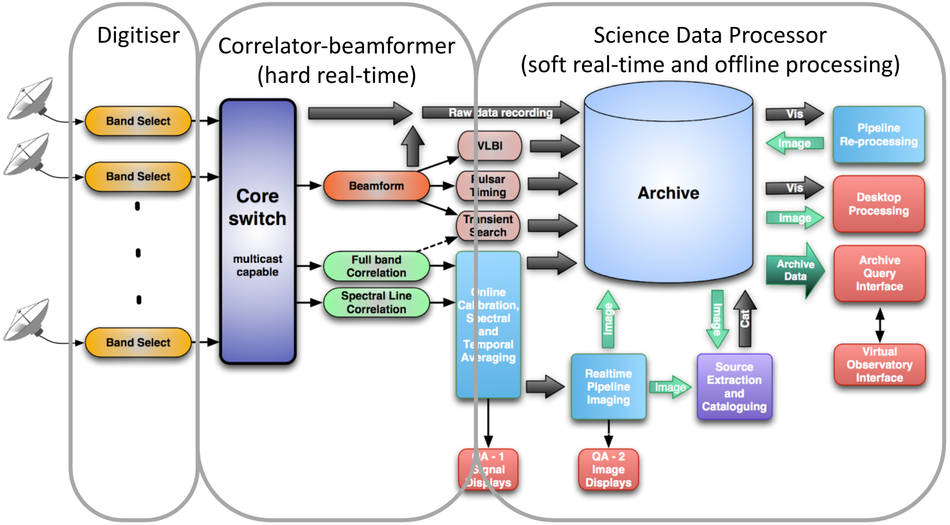
Re-cap

- MeerKAT Radio Telescope is getting an extension.
- The correlator is the part which digitally combines signals from all the individual antennas.

Re-cap

- MeerKAT Radio Telescope is getting an extension.
- The correlator is the part which digitally combines signals from all the individual antennas.
- The correlator being built for MeerKAT Extension is (possibly) the world's first real-time correlator written (almost) entirely in Python.

Correlator Context



A look inside...

Need for testing

Lots of functionality to verify

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you’ve built meet spec?

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements
  - graphs

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements
  - graphs
  - tables

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements
  - graphs
  - tables
  - etc.

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements
  - graphs
  - tables
  - etc.
- To show evidence of meeting spec.

Systems Engineering

- Formal discipline for designing and managing complex systems over life cycle
- Started at Bell Labs in the 40s, influenced heavily by military procedures
- Typically (relatively) document-heavy
- Qualificatyion report - Does the system you've built meet spec?
- Pytest's default output - not good enough. We need
  - measurements
  - graphs
  - tables
  - etc.
- To show evidence of meeting spec.
- Very rigorous, quite boring to do manually. Fortunately, pytest can still help!

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

What do we test?

A non-exhaustive list:

- Signal Processing correctness

What do we test?

A non-exhaustive list:

- Signal Processing correctness
- Response to control inputs

What do we test?

A non-exhaustive list:

- Signal Processing correctness
- Response to control inputs
- Correct reporting of internal state

What do we test?

A non-exhaustive list:

- Signal Processing correctness
- Response to control inputs
- Correct reporting of internal state
- Fault handling

What do we test?

A non-exhaustive list:

- Signal Processing correctness
- Response to control inputs
- Correct reporting of internal state
- Fault handling
- Ultimately anything that the specifications require

How?

- Don’t re-invent the wheel (or the test framework)

How?

- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:

How?

- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:
    - Representative test system

How?

- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:
  - Representative test system
  - Mechanism to communicate with this system

How?

- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:
  - Representative test system
  - Mechanism to communicate with this system
  - Controlled, deterministic inputs (and environment if that’s important)

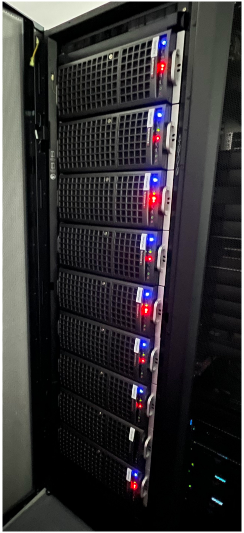
How?

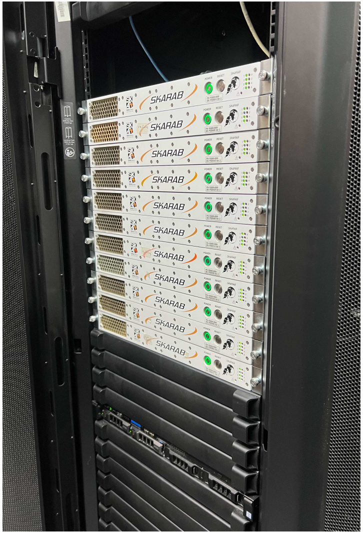
- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:
  - Representative test system
  - Mechanism to communicate with this system
  - Controlled, deterministic inputs (and environment if that’s important)
  - Measurement of outputs

How?

- Don’t re-invent the wheel (or the test framework)
- What you do need to invent:
  - Representative test system
  - Mechanism to communicate with this system
  - Controlled, deterministic inputs (and environment if that’s important)
  - Measurement of outputs
  - Standard or ideal against which to compare

Our examples





Using pytest

- Pytest fixtures do the hard work

Using pytest

- Pytest fixtures do the hard work
  - Start and control a correlator (device under test)

Using pytest

- Pytest fixtures do the hard work
  - Start and control a correlator (device under test)
  - Start and control a digitiser simulator (input)

Using pytest

- Pytest fixtures do the hard work
    - Start and control a correlator (device under test)
    - Start and control a digitiser simulator (input)
    - Receive correlator output (measurement)

Using pytest

- Pytest fixtures do the hard work
    - Start and control a correlator (device under test)
    - Start and control a digitiser simulator (input)
    - Receive correlator output (measurement)
    - Generate documentation

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process
- Ideally done early - how will you prove that you’ve met spec?

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process
- Ideally done early - how will you prove that you've met spec?
- Normally a separate test procedure document

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process
- Ideally done early - how will you prove that you’ve met spec?
- Normally a separate test procedure document
- Our approach - let the test script be the procedure (interleave prose with code)

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process
- Ideally done early - how will you prove that you've met spec?
- Normally a separate test procedure document
- Our approach - let the test script be the procedure (interleave prose with code)
- Thereby minimising drift

Procedures and Reports

- Detailed test procedures are an aspect of the systems engineering process
- Ideally done early - how will you prove that you’ve met spec?
- Normally a separate test procedure document
- Our approach - let the test script be the procedure (interleave prose with code)
- Thereby minimising drift
- The same fixture used for procedures, produces reports

Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

Linearity test

```python
"""CBF linearity test."""

async def test_linearity(
    correlator: CorrelatorRemoteControl,
    receive_baseline_correlation_products:
        BaselineCorrelationProductsReceiver,
    pdf_report: Reporter,
) -> None:
    receiver = receive_baseline_correlation_products

    pdf_report.step("Select a range of CW scales for testing.")
    cw_scales = [0.5**i for i in range(10)]
    pdf_report.detail(f"CW scales: {cw_scales}")
```

Linearity test

```python
    pdf_report.step("Select a channel and compute the channel center frequency for the D-sim.")
    sel_chan_center = receiver.n_chans // 3
    channel_frequency = sel_chan_center * (receiver.bandwidth / receiver.n_chans)
    pdf_report.detail(
        f"Channel {sel_chan_center} selected, with center frequency " + f"{channel_frequency/1e6:.2f} MHz."
    )

    pdf_report.step("Set EQ gain.")
    gain = compute_tone_gain(receiver=receiver, amplitude=max(cw_scales), target_voltage=110)

    pdf_report.detail(f"Setting gain to: {gain}")
    await correlator.product_controller_client.request("gain-all", "antenna_channelised_voltage", gain)
```

Linearity test

```
    base_corr_prod = await sample_tone_response(
        rel_freqs=sel_chan_center,
        amplitude=cw_scales,
        receiver=receiver,
    )

    linear_scale_result = base_corr_prod[:, sel_chan_center]
```

Linearity test

```
# Normalise and compute the effective received voltage value (from power) for comparison to the requested value.
linear_test_result = np.sqrt(linear_scale_result / np.max(linear_scale_result))

pdf_report.step("Compute RMS Voltage.")
rms_voltage = np.sqrt(np.max(linear_scale_result) / receiver.n_spectra_per_acc)
pdf_report.detail(f"RMS voltage: {rms_voltage:.3f}.")

pdf_report.step("Compute Mean Square Error (MSE).")
mse = np.square(cw_scales - linear_test_result).mean()
pdf_report.detail(f"MSE is: {mse}")
```

Linearity test

```python
# Generate plot with reference
labels = [f"$2^{{-{i}}}$" for i in range(len(cw_scales))]
title = "Power relative to input CW level"
xticks = np.arange(len(cw_scales))
fig = Figure()
ax = fig.subplots()
ax.plot(20 * np.log10(cw_scales), label="Reference")
with np.errstate(divide="ignore"):  # Avoid warnings when the value is zero
    ax.plot(20 * np.log10(linear_test_result), label="Measured")
ax.set_title(title)
ax.set_xlabel("CW Scale")
ax.set_ylabel("dB")
ax.legend()
ax.set_xticks(xticks)
ax.set_xticklabels(labels)
pdf_report.figure(fig)
```

Example Report

1 Requirements Verified

Requirement	Tests verifying requirement

2 Test Configuration

Test suite	
Tester	Unknown
Test Suite Git Info	8f798b2
Correlator	
Image	harbor.sdp.kat.ac.za/cbf/katgpucbf@sha256: 78dbc403913d1185b11c7a41e651f9d52d12381fad36d2e814264917053d556c
Version	0.1.dev2396+g2674ab5.d20220816
Product Controller	
Image	sdp-docker-registry.kat.ac.za:5000/katsdpcontroller@sha256: ee2e511ae714940ccc703468178147fb5ab9b75740d9710ccc31a641bd743763
Version	katsdpcontroller-0.1.dev2602+head.10dc7d5

3 Hosts

cbfgpu04.sdpdyn.kat.ac.za cbfgpu05.sdpdyn.kat.ac.za	
System	
CPU	AMD EPYC 7313P 16-Core Processor

Example Report

4 Correlators

4.1 Configuration 1: 8n856M8k

4 antennas, 8192 channels, L-band, 0.5s integrations, 4 dsims.

Product controller	lab5.sdp.kat.ac.za
DSim 0	cbfgpu05.sdpdyn.kat.ac.za
DSim 1	cbfgpu05.sdpdyn.kat.ac.za
DSim 2	cbfgpu05.sdpdyn.kat.ac.za
DSim 3	cbfgpu05.sdpdyn.kat.ac.za
F-engine 0	cbfgpu04.sdpdyn.kat.ac.za
F-engine 1	cbfgpu04.sdpdyn.kat.ac.za
F-engine 2	cbfgpu04.sdpdyn.kat.ac.za
F-engine 3	cbfgpu04.sdpdyn.kat.ac.za
XB-engine 0	cbfgpu08.sdpdyn.kat.ac.za
XB-engine 1	cbfgpu08.sdpdyn.kat.ac.za
XB-engine 2	cbfgpu08.sdpdyn.kat.ac.za
XB-engine 3	cbfgpu08.sdpdyn.kat.ac.za

6 Detailed Test Results

6.1 Linearity [8192-l]

Test that baseline Correlation Products are linear when input CW is scaled.

6.1.1 Verification method

Verify linearity by checking the channelised output scales linearly with a linear change to the CW input amplitude.

6.1.2 Requirements Verified

None

6.1.3 Results

Outcome: PASSED Test start time: 15:28:25 Duration: 40.058 s seconds

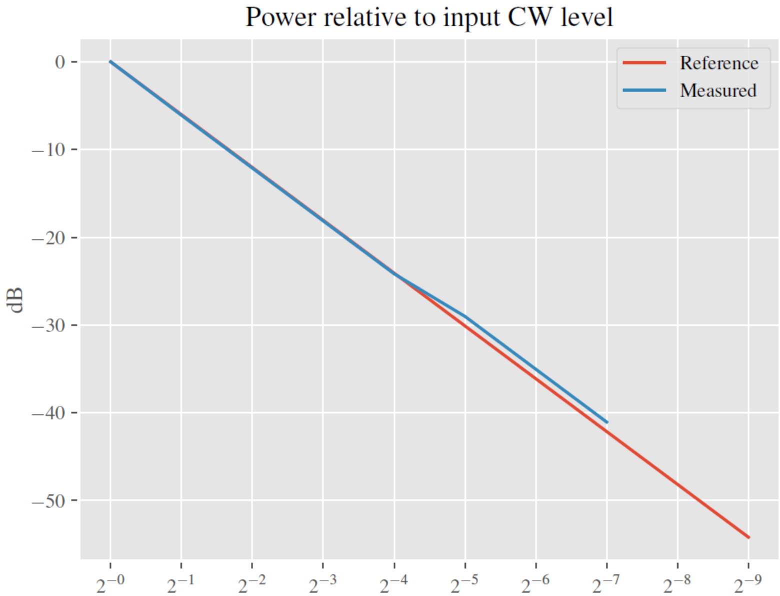
Test Configuration	
Correlator	Configuration 1: 8n856M8k

Example Report

6.1.4 Procedure

Capture channelised data for various input CW scales and check linearity.	
Select a range of CW scales for testing.	
2.505 s	CW scales: [1.0, 0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0.00390625, 0.001953125]
Select a channel and compute the channel center frequency for the D-sim.	
2.505 s	Channel 2730 selected, with center frequency 285.26 MHz.
Set EQ gain.	
2.506 s	Setting gain to: 0.0033635029354207435
Compute RMS Voltage.	
8.501 s	RMS voltage: 113.000.
Compute Mean Square Error (MSE).	
8.501 s	MSE is: 6.766165793758939e-06

Example Report



Outline

- Introduction
- MeerKAT (Extension) Correlator
- Systems engineering
- System level tests
- Example test
- Winding up

Benefits

- Finding problems before they go out into the wild

Benefits

- Finding problems before they go out into the wild
    - Sometimes this involves confronting unpleasant realities

Benefits

- Finding problems before they go out into the wild
  - Sometimes this involves confronting unpleasant realities
  - But it has helped us in the past

Benefits

- Finding problems before they go out into the wild
  - Sometimes this involves confronting unpleasant realities
  - But it has helped us in the past
  - and it continues to do so

Benefits

- Finding problems before they go out into the wild
  - Sometimes this involves confronting unpleasant realities
  - But it has helped us in the past
  - and it continues to do so
- Value to collaborators (and potential collaborators)

Benefits

- Finding problems before they go out into the wild
  - Sometimes this involves confronting unpleasant realities
  - But it has helped us in the past
  - and it continues to do so
- Value to collaborators (and potential collaborators)
  - Run a mini-version on your laptop using Docker

Benefits

- Finding problems before they go out into the wild
  - Sometimes this involves confronting unpleasant realities
  - But it has helped us in the past
  - and it continues to do so
- Value to collaborators (and potential collaborators)
  - Run a mini-version on your laptop using Docker
  - Play with code, understand how changes affect output

Benefits

- Finding problems before they go out into the wild
    - Sometimes this involves confronting unpleasant realities
    - But it has helped us in the past
    - and it continues to do so
- Value to collaborators (and potential collaborators)
    - Run a mini-version on your laptop using Docker
    - Play with code, understand how changes affect output
    - Increases visibility and transparency in scientific process

Conclusion

Testing - it's not just for little code snippets.
You can test big things as well!

science & innovation
Department:
Science and Innovation
REPUBLIC OF SOUTH AFRICA

NRF National Research Foundation | SARAO South African Radio Astronomy Observatory

Think any of this is cool? Come and work with us!

James Smith

DSP Engineer

Email: jsmith@sarao.ac.za

Save the trees, please don't print the qualification reports!

www.sarao.ac.za

The South African Radio Astronomy Observatory (SARAO) is a National Facility managed by the National Research Foundation and incorporates all national radio astronomy telescopes and programmes.