ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR004

LEVERAGING SOFTWARE SIMULATORS IN SKA DISH LMC DEVELOPMENT AND TESTING

S. N. Twum*, SKA Observatory, Jodrell Bank, United Kingdom
A. Lethole, M. Nzama, B. Ojur, N. Padavattan, D. Petrie, Y. Reddi, U. C. Silere, A. Venter
South African Radio Astronomy Observatory, Cape Town, South Africa

Abstract

The Square Kilometre Array (SKA) project employs the TANGO Controls framework to manage its telescopes. Within SKA MID, the Dish Local Monitoring and Control (LMC) software integrates dish sub-components into the overall control system. Because hardware availability is often limited, Dish LMC development and validation are heavily based on software simulators derived from Interface Control Documents (ICDs) and system-level requirements. These simulators emulate the behaviour of the respective devices for requirement verification. Their use has enabled iterative testing in both staging and integration environments, accelerated development, and reduced risks ahead of hardware integration. This paper presents the design and implementation of Dish LMC simulators, their role in supporting system-level validation and advancing software maturity.

THE SKA CONTROL SYSTEM - DISH LMC'S ROLE

The SKA project is a global endeavor to construct the world's largest radio telescope: SKA MID in South Africa (operating between 350 MHz – 15.4 GHz) and SKA LOW in Australia (operating between 50 – 350 MHz). SKA MID will comprise 197 Gregorian offset dishes, including 64 MeerKAT dishes and 133 SKA dishes. In parallel, SKA LOW will employ 131,072 dipole antennas grouped into 512 stations. Together, these telescopes will cover a collecting area of one square kilometre [1].

A distributed Tango control system provides remote access to the telescope subsystems [2], as well as other hardware on site. The dish subsystem implements a Tango interface that aggregates sensor data from its subcomponents to communicate the overall dish state. Most monitored subcomponents expose a Tango interface, while a few implement either OPCUA, SNMP, or other proprietary communication protocols.

Dish LMC provides master control and aggregated monitoring of the dish subsystem shown in Fig. 1. Dish LMC's architecture, shown in Fig. 2, includes a power manager, single-pixel feed controller (SPF), single-pixel feed receiver (SPFRx) controller, Band 5 down converter (B5DC), dish structure manager (DS), and dish manager [3, 4]. The development of Dish LMC is guided by ICDs and system requirements. However, the limited availability of hardware has necessitated the use of software simulators to accelerate development, mitigate risks, and validate requirements early.

* samuel.twum@skao.int

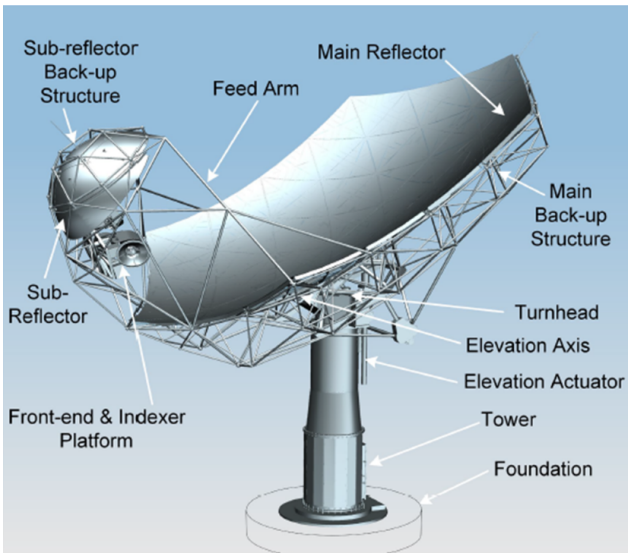

Figure 1: SKA MID Dish elements.

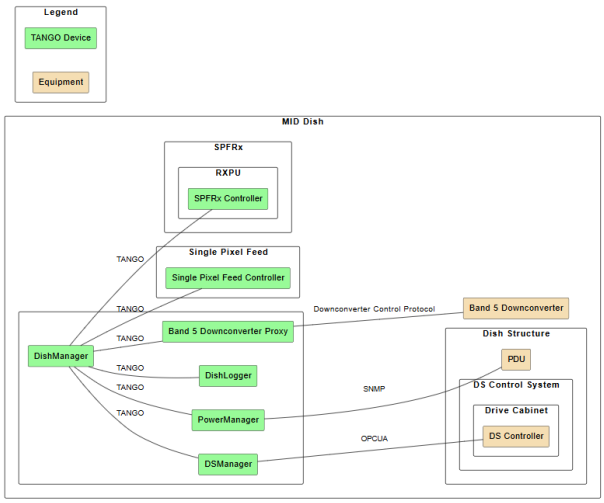

Figure 2: Dish LMC architecture overview.

DISH LMC SIMULATION APPROACH

The various software components comprising the Dish LMC are developed by different teams who are not contracted to provide simulators for their product. Consequently, significant effort is invested in building a simulator that represents the hardware. The Dish LMC simulators are designed to reproduce the behaviour of hardware subcomponents based on ICDs, ensuring that software integration aligns

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR004

with formal specifications. Rather than physically modelling complex dynamics, the simulators provide API-level consistency and protocol compliance. For critical functions, such as the dish movement, simulators implement feedback control loops and state behaviour. The ICDs and requirements are the source of truth for verifying the correctness of the simulations of the actual interfaces. These simulators allow acceptance testing, scenario-driven requirement verification, and flexibility during integration. By default, the simulators are deployed with Dish LMC using configurable Helm parameters, which enable selective replacement of simulators with actual hardware as it becomes available. The example snippet below shows a helm yaml configuration to deploy Dish LMC along with its simulators as shown in Fig. 3).

```
...
# SPF, SPFRx, DSC simulators
ska-mid-dish-simulators:
  enabled: true
  deviceServers:
    spfdevice:
      enabled: true
    spfrxdevice:
      enabled: true
  dsOpcuaSimulator:
    enabled: true

# B5DC simulator
ska-mid-dish-dcp-lib:
  enabled: true
  b5dcSimulator:
    enabled: true
```

DISH LMC SIMULATOR IMPLEMENTATIONS

Currently, there are four simulators for the SPF, SPFRx, B5DC, and DSC, respectively. The SPF, SPFRx and DSC simulators are maintained in `ska-mid-dish-simulators` [5] and the B5DC simulator is maintained in `ska-mid-dish-dcp-lib` [6].

SPF and SPFRx Simulator

These two devices are lightweight Python Tango devices implementing expected outcomes with minimal embedded behaviour. Both SPF and SPFRx devices control lower-level Tango devices that abstract different hardware components [5].

B5DC Simulator

The B5DC library provides a UDP server that emulates the behaviour of the B5DC device [6]. This simulator replicates the device's functionality by creating an internal dictionary

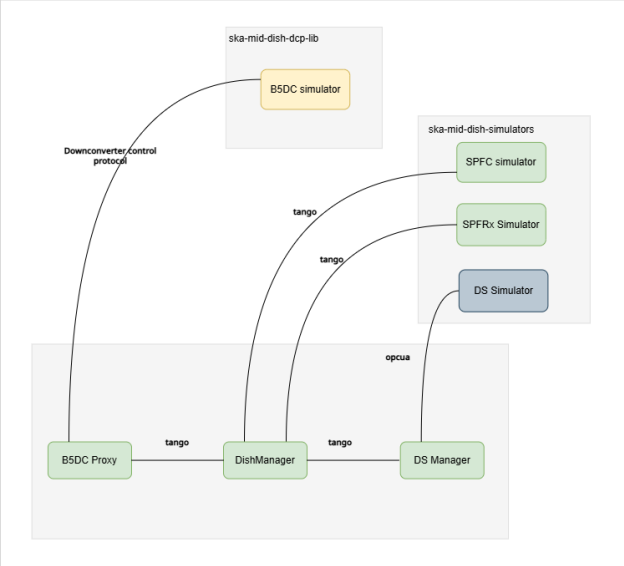

Figure 3: Dish LMC software and simulators: current status.

that models the B5DC register bank. A UDP client, implementing the Down Converter Protocol (DCP), can access this virtual register bank to read and write values, mirroring interaction with the physical B5DC device. The virtual register addresses correspond to those parsed from the ICD.

DSC Simulator

This simulator implements an OPC-UA device closely approximating states and modes, and motion of the dish [5]. The azimuth and elevation motion are simulated by implementing cascaded velocity and position feedback loops. The dynamics was split into position and velocity loops so that saturation limits could be put in place to emulate the rate limits of the physical dish. The inner velocity loop was implemented as a first-order closed-loop system with a time-constant that represents the specifications of the physical dish dynamics. The outer position loop employs a simple proportional gain feedback mechanism. This gain is chosen based on assessment of the closed-loop dynamics of the physical dish. Continuous tracking is handled through a reference table of time-stamped azimuth and elevation points, which are fed into the servo loop controller to reproduce realistic pointing trajectories with delays, settling behaviour, and tolerance thresholds similar to the physical dish. Dish LMC leverages a shared library to interface with the OPC server which is also used by the DISQ qualification tool [7].

Together, these simulators establish a comprehensive testing environment, ensuring that Dish LMC can be exercised thoroughly before hardware integration. The example snippet below shows Dish LMC deployed to connect to the actual interfaces implementing the DSC and SPFRx.

```
ska-mid-dish-manager:
  dishmanager:
    # spfrx hardware TRL
    spfrx:
```

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR004


```
      fqdn: "SKA005/spfrxpu/controller"

ska-mid-dish-ds-manager:
  dishstructuremanager:
    # DSC opcua device address
    dsc:
      fqdns: ["opc.tcp://127.0.0.1:8080/"]
    ds_opcua_namespace: "http://namespace"

ska-mid-dish-simulators:
  enabled: true
  deviceServers:
    # spfrx simulator DISABLED
    spfrxdevice:
      enabled: false
    # opcua simulator DISABLED
    dsOpcuaSimulator:
      enabled: false
    # spf simulator enabled
    spfdevice:
      enabled: true
```

TESTING STRATEGY, INTEGRATION AND QUALIFICATION

Integration teams deploy and test Dish LMC as part of SKA MID product integration. In staging environments, Dish LMC is validated with all simulators enabled, verifying basic subsystem control. As integration progresses, specific simulators are replaced with hardware while maintaining others, ensuring incremental validation and seamless transitions. This multi-stage approach (see Fig. 4) supports continuous integration and regression testing, highlighting behavioural differences between simulators and hardware. The flexibility to disable simulators selectively has accelerated verification, improved software maturity, and reduced risks associated with hardware delays.

LIMITATIONS, BENEFITS, AND FATE OF THE SIMULATORS

The use of simulators in the Dish LMC development and testing has presented both advantages and challenges. While they offer a controlled environment for early-stage development and iterative testing, they inherently carry limitations. One key limitation is the inherent abstraction from the actual hardware; despite efforts to approximate real-world behavior, simulators cannot perfectly replicate all physical nuances, transient states, or unforeseen interactions that might occur with the actual sub-components. This can lead to discrepancies between simulated and real-world performance, requiring further validation with physical hardware. Furthermore, the development and maintenance of these simulators require significant effort and resources, especially as ICDs and

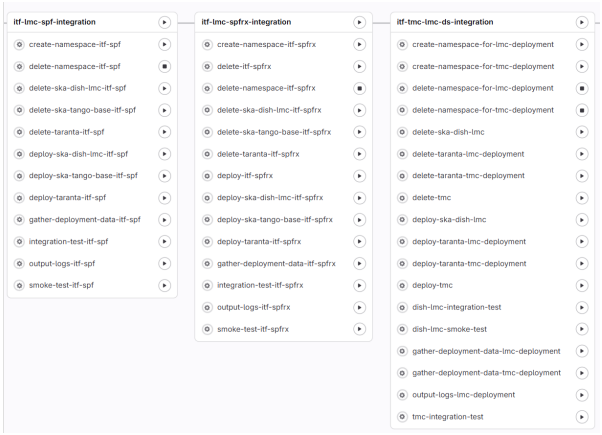

Figure 4: Snapshot of Dish LMC hardware integration from manual pipeline jobs.

hardware interfaces evolve. There is also the risk of "simulation drift," where the simulator's behavior might subtly diverge from the actual hardware over time if not rigorously updated and validated.

Despite these limitations, the benefits of employing these simulators have been substantial. They enable parallel development of software and hardware, significantly accelerating the overall project timeline. By providing a stable and readily available testing environment, simulators facilitate continuous integration and early detection of defects, reducing the cost and complexity of bug fixes later in the development cycle. They also allow for the testing of edge cases and fault conditions that would be difficult, expensive, or even risky to reproduce with actual hardware. The ability to disable specific simulators through Helm deployment parameters further enhances testing flexibility, allowing for targeted verification with available physical components. This has been particularly beneficial in driving the maturity of the Dish LMC software and in contributing to qualification and verification procedures.

Looking ahead, the fate of these simulators is likely to evolve alongside the project's progression. As more physical hardware becomes available and integrated, the role of the simulators may shift from primary testing tools to supplementary resources for specific scenarios. They will continue to be invaluable for regression testing of new software releases and for exploring future design iterations without impacting operational telescopes. However, as the system matures, the emphasis will naturally move towards extensive testing with the actual hardware. Nevertheless, the initial investment in these simulators has proven critical in de-risking the Dish LMC development and will likely remain a crucial component of the SKA MID Telescope's long-term maintenance and upgrade strategy.

ACKNOWLEDGEMENTS

The authors would like to thank the SKA Observatory (SKAO) and the South African Radio Astronomy Observatory (SARAO) for their support in the development and

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR004

testing of the Dish LMC software. We acknowledge the contributions of the integration teams whose efforts and collaborative feedback were invaluable in shaping the simulators and integration workflows.

REFERENCES

[1] P. E. Dewdney, *et al.*, "The Square Kilometre Array", *Proc. IEEE*, vol. 97, no. 8, pp. 1482–1496, Aug. 2009. doi:10.1109/jproc.2009.2021005

[2] S. Vrcic and T. Juerges, "SKA telescope control system design and status", in *Proc. SPIE*, vol. 13101, p. 1310103, Jul. 2024. doi:10.1117/12.3018723

[3] S. Riggi, *et al.*, "The SKA Dish Local Monitoring and Control System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 508–512, doi:10.18429/JACoW-ICALEPCS2017-TUPHA050

[4] Dish LMC Documentation, https://developer.skao.int/projects/ska-dish-lmc/en/8.4.1/.

[5] Dish LMC Simulators, https://gitlab.com/ska-telescope/ska-mid-dish-simulators/-/tree/5.0.0?ref_type=tags

[6] Band 5 Down-converter Communication Library, https://gitlab.com/ska-telescope/ska-mid-dish-dcp-lib/-/tree/0.0.5?ref_type=tags

[7] U. K. Pedersen, *et al.*, "Field deployment and iterative enhancement of the dish structure qualification (DiSQ) software for SKA-Mid", unpublished.