Assembly, Integration, and Verification of the Cherenkov Telescope Array Observatory Data Processing and Preservation System

Volodymyr Savchenko, on behalf of the CTAO DPPS team and SUSS AIV team

[1]*EPFL, Lausanne, Switzerland;* `Volodymyr.Savchenko@epfl.ch`

Abstract.
The Cherenkov Telescope Array Observatory (CTAO) is the next-generation very-high energy gamma-ray observatory currently under construction. The bulk of the CTAO data will be processed by the Data Processing and Preservation System (DPPS), yielding high-level data ready for elaboration by astronomers. Assembly, Integration, and Verification (AIV) is a key element of the DPPS development lifecycle, integrating its subsystems into a cohesive system, and verifying that it meets its requirements and quality standards during development and deployment, as well as creating the means to ensure its long-term operability and maintainability.

DPPS AIV embraces the synergy of dynamic Continuous Delivery and rigorous Requirements Verification workflows, built upon DevOps principles enabling reproducible deployments closely aligned between development, testing, pre-production, and production environments as well as reliable fail-over between different sites of the distributed production infrastructure.

1. CTAO and DPPS

The Cherenkov Telescope Array Observatory (CTAO) represents the next generation of ground-based gamma-ray astronomy. Currently under construction at two sites – La Palma, Spain (North) and Paranal, Chile (South), the observatory aims to detect high-energy photons in the energy range from around 20 GeV to 300 TeV with unprecedented sensitivity, angular resolution, and energy resolution.

The bulk of the CTAO data will be processed by the Data Processing and Preservation System (DPPS) [1], yielding high-level data ready for elaboration by astronomers. DPPS enables **Data Preservation** for decades of the observatory lifetime with multi-site redundancy, **High-Volume Data-Driven Processing** by executing complex data-driven workflows converting raw telescope data into science-ready data products.

The DPPS the following management systems: **WMS** – Workload Management System, based on DIRAC and CVMFS (Tsaregorodtsev et al. 2008; Blomer et al.) for workload orchestration and distributed computing ; **BDMS** – Bulk Data Management System, based on Rucio (Barisits et al. 2019), for data lifecycle management across multiple geographically distributed data centers; and **Ops** – for deployment, and ob-

[1]`https://gitlab.cta-observatory.org/cta-computing/dpps/dpps/`

servability of DPPS services, based on Kubernetes, Helm, and Grafana observability stack.

DPPS orchestrates data processing pipelines, grouped into four subsystems: **CalibPipe** handles telescope and ancillary instrument calibration, **DataPipe** performs event reconstruction, **SimPipe** produces Monte Carlo simulations, and **QualPipe** ensures data quality.

2. DPPS deployment strategy

The deployment strategy lays the foundation of DPPS verification and release lifecycle by defining the way DPPS is brought to life in various environments. It is built on the idea of **reproducible deployment**, allowing to easily create full copies of the DPPS system in any standard Kubernetes cluster.

DPPS Helm charts enable reproducible deployments in any standard Kubernetes cluster. The DPPS system includes its components as Helm chart dependencies. All DPPS Helm charts include test versions of all dependent services.The charts also take care of bootstrapping, upgrades (including migrations), backups and restore.

DPPS CWL pipelines (Amstutz et al. 2016) are used to define containerized data processing pipelines executed by DPPS WMS since Kubernetes native Job management is not currently suitable for defining complex data processing workflows. DPPS pipelines can be executed in any CWL-compliant execution environment, local or distributed.

All DPPS release artifacts are versioned and stored in an OCI-compliant Harbor (`https://goharbor.io/`) container registry. All DPPS components can be deployed in multiple **deployment environments**. **Local development** and **CI/CD ephemeral environments** are ephemeral environments created on-demand in developer workstations. **Staging**, **Pre-production**: and **Production** are persistent deployment environments managed as GitOps repositories.

Observability is a key aspect of the DPPS deployment strategy, allowing to design, monitor, and verify DPPS deployments at every stage of the deployment lifecycle, producing standardized metrics, logs, and alerts, which can be interrogated and analyzed in the same way as full-scale pre-production and production deployments.

3. Release Lifecycle and Verification Strategy

DPPS embraces a rapid, trunk-based development cycle, with relatively long sprints (about eight weeks) each concluding with a DPPS system release. AIV is central in this process by providing the means to verify DPPS components and system as a whole at every stage of development and deployment.

CTAO, the stakeholder of DPPS, is defining high-level requirements and use cases in the Jama requirements management system. DPPS AIV ensures that every DPPS release is compliant with its requirements by executing automated tests in the CI/CD pipelines upon every code change.

Given that containerized deployment can be easily ported to any standard environment, the main deployment challenge is to ensure that all **system interfaces** are correctly implemented and configured. In addition to scenario-driven integration tests, we test configurability by introducing randomized configuration of interfaces.

Large-scale performance tests are only performed in **pre-production or production** infrastructure, first developed just like other automated tests, and later configured for pre-production environment. These will continue to be used to verify ongoing compliance of live DPPS deployment with its requirements.

All DPPS code repositories undergo static code analysis using SonarQube [2] to ensure code quality, maintainability, and security. AIV also collects and verifies information about third-party dependencies.

4. CTAO AIV toolkit

In order to efficiently manage the complexity of DPPS AIV processes and tools, we developed a custom toolkit[3] to automate and streamline various AIV tasks. Originally developed to support DPPS, it is now also adopted by another CTAO software project, SUSS. The toolkit includes a **Python Package**, **CI/CD Pipeline Templates** for consistent testing and deployment across projects, a **pytest plugin**: for annotating tests with requirement traceability information, an **AIV Toolkit Helm Chart** for deploying services facilitating testing, and a **pre-commit Hook**.

Before developing our own toolkit, we evaluated existing solutions for setting up a Helm-based development environment, such as skaffold (`https://skaffold.dev/`), and chartpress (`https://chartpress.readthedocs.io/`), but found that they did not fully meet our specific needs, and are difficult to extend to support requirement traceability.

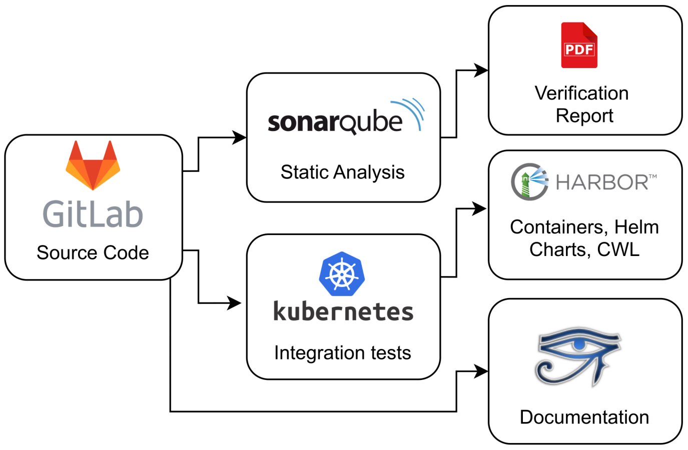

Figure 1. An example of a DPPS AIV CI/CD pipeline using CTAO AIV toolkit. This pipeline performs static code analysis, deploys fully-functional DPPS instance, runs integration tests, and generates verification reports.

[2] `https://sonar-ctao.zeuthen.desy.de/`

[3] `https://gitlab.cta-observatory.org/cta-computing/common/aiv-toolkit`

To automate AIV tasks not triggered by code changes, we developed an **AIV Bot**[4], taking care of **release issue management**, **dependency updates**, and integration between Gitlab issue code with **Jama requirements** management system. The AIV bot also contains experimental functionality for generative LLMs to assist with some of its tasks, such as drafting issue descriptions or summarizing changes and comparing them to requirements.

5. LLMs in DPPS AIV

The advent of LLMs has introduced both opportunities and risks to the CTAO software stack. While freeing from certain tasks, it also introduces suboptimal or incorrect code that may pass superficial reviews. Since DPPS AIV is tasked with verifying software quality, it plays a crucial role in mitigating AI-related risks. For example, AI can assist in generating test cases based on requirements, or in analyzing test results to identify patterns or anomalies that may indicate deeper issues.

6. Challenges, Conclusions, and Outlook

Since the first DPPS release in February 2025, DPPS has undergone six minor releases, each verified against a progressively larger set of requirements. Only six months after the first release, the DPPS successfully made first pre-production deployment, and underwent several tests on real data acquired from on-site telescopes, demonstrating that DPPS AIV strategy is effective in ensuring that DPPS meets its requirements and quality standards before deployment.

As DPPS development progresses the entire system becomes increasingly complex, with more components, interfaces, and requirements to manage. The current DPPS version comprises almost a hundred containers, dozens of Helm charts, and hundreds of automated tests. The AIV team is continuously adapting to this growing complexity, helping to ensure that the developers can focus on building new features while maintaining confidence in DPPS releases being production-ready.

Acknowledgments. This work was conducted in the context of the CTAO DPPS and SUSS AIV teams. The author thanks the DPPS and SUSS team members for fruitful discussions and feedback. The content of this paper was produced with limited support of several LLMs for drafting and editing assistance.

References

Amstutz, P., et al. 2016, Common workflow language, v1.0. URL `https://doi.org/10.6084/m9.figshare.3115156.v2`

Barisits, M., et al. 2019, Computing and Software for Big Science, 3. URL `http://dx.doi.org/10.1007/s41781-019-0026-3`

Blomer, J., et al.The CernVM File System (CVMFS). URL `https://github.com/cvmfs/cvmfs`

Tsaregorodtsev, A., et al. 2008, Journal of Physics: Conference Series, 119, 062048. URL `https://api.semanticscholar.org/CorpusID:5719579`

[4]`https://gitlab.cta-observatory.org/cta-computing/dpps/aiv/gitlab-operator/`