THE ARRAY CONTROL AND DATA ACQUISITION SYSTEM OF THE CHERENKOV TELESCOPE ARRAY

I. Oya*, E. Antolini, M. Füßling, J. A. Hinton, A. Mitchell, S. Schlenstedt
CTAO gGmbH, Heidelberg, Germany
L. Baroncelli, A. Bulgarelli, V. Conforti, N. Parmiggiani, INAF - O.A.S. Bologna, Italy
J. Borkowski, CAMK, Torun, Poland
A. Carosi, J. Jacquemier, G. Maurin
LAPP, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, France
J. Colomé, IEEC-CSIC, Spain
C. Hoischen, Universität Potsdam, Germany
D. Lebout, Université Paris-Sud, CNRS, INRIA, Université Paris-Saclay, Orsay, France
E. Lyard, R. Walter, University of Geneva, Switzerland
D. Melkumyan, K. Mosshammer, I. Sadeh, T. Schmidt, P. Wegner, DESY, Zeuthen, Germany
U. Schwanke, Humboldt-Universität zu Berlin, Germany
J. Schwarz, INAF - Osservatorio Astronomico di Brera, Italy
G. Tosti, University of Perugia, Italy
for the CTA Observatory†

Abstract

The Cherenkov Telescope Array (CTA) project is the initiative to build the next-generation gamma-ray observatory. With more than 100 telescopes planned to be deployed at two sites, CTA is one of the largest astronomical facilities under construction. The Array Control and Data Acquisition (ACADA) system will be the central element of on-site CTA Observatory operations. The mission of the ACADA system is to manage and optimize the telescope array operations at each of the CTA sites. To that end, ACADA will provide all necessary means for the efficient execution of observations, and for the handling of the several Gb/s of data generated by each individual CTA telescope. The ACADA system will contain a real-time analysis pipeline, dedicated to the automatic generation of science alert candidates as data are being acquired. These science alerts, together with external alerts arriving from other scientific installations, will permit ACADA to modify ongoing observations at sub-minute timescales in order to study high-impact scientific transient phenomena. This contribution describes the challenges, architecture, design principles, and development status of the ACADA system.

INTRODUCTION

The Cherenkov Telescope Array (CTA) is an initiative to build the next-generation atmospheric Cherenkov gamma-ray observatory [1]. The CTA Observatory (CTAO) will consist of two facilities, one in the Southern (close to the Paranal Observatory, Chile) and the other in the Northern Hemisphere (at the Roque de Los Muchachos Observatory, La Palma, Spain). The two sites will contain dozens of telescopes of different sizes – Large, Medium, and Small-Sized Telescopes (LSTs, MSTs, SSTs) –, constituting one of the largest astronomical installations under development.

The observations of the CTA installation will be coordinated by the Array Control and Data Acquisition (ACADA) System. Two major software systems [2] of CTA are the Data Processing and Preservation Systems (DPPS), which will be hosted in a set of offsite data centres, and the Science User Support System (SUSS), which will be deployed in the CTA Science Data Management Centre in Zeuthen, Germany. SUSS will, among other functions, make processed data and science tools available to end users, as well as provide the interface to create and submit CTA observation proposals.

This contribution describes the ACADA system. After introducing the mission of ACADA and its main requirements, we present an overview of the ACADA system architecture, followed by the most significant design principles of the system. We close with a summary of the development status of the ACADA system.

ACADA SYSTEM MISSION

The ACADA system comprises all software responsible for the control and data acquisition of telescopes and the additional devices responsible for array calibration and environment monitoring at each of the CTA sites; it plays the role of the supervisory control and data acquisition (SCADA) system. ACADA is also responsible for the efficient execution of pre-scheduled observations and those triggered by science alerts – which allows CTA to respond on sub-minute timescales and observe interesting transient phenomena such as gamma-ray bursts [3]. These science alerts can be triggered externally by other scientific installations, or internally within ACADA thanks to a dedicated

* igor.oya@cta-observatory.org
† http://www.cta-observatory.org

analysis running in real-time (the Science Alert Generation Pipeline, see below). The ACADA system also provides the user interface for the site operators and astronomers.

MAIN REQUIREMENTS FOR THE ACADA SYSTEM

The CTA team has identified and documented about 200 individual requirements for the ACADA system. A selection of the most important ones is given in the following:

- **Operation Modes**: ACADA must have both a Science Operations Mode, in which only science observations may take place, and a Technical Operations Mode. In the latter science observations are not possible but monitoring, maintenance and engineering capabilities remain available.
- **Subarray Operation:** ACADA must permit the operation of at least eight independently operable subarrays at the same time, formed from distinct subsets of available telescopes at a given site.
- **Software Restart Efficiency:** A full restart of ACADA software must be possible within 2 minutes, with full monitoring and alarm functionality of all available Array Elements available within 5 minutes.
- **Transient or variable source detection:** ACADA must conduct a full field-of-view search for transient and/or time-variable phenomena. Based on configurable thresholds, this search must be used to detect candidate science alerts.
- **Issuing Alerts:** ACADA must be capable of issuing the alert to the external world within 10 seconds of the detection of a candidate alert by the alert generation pipeline.
- **Users:** ACADA shall be designed to be operated by an operator and a support astronomer located in a control room.
- **Incoming target of opportunity information:** ACADA must interface to international networks for astrophysical transients; receiving, processing and filtering incoming information within 5 seconds of receipt in order to potentially trigger transient observations.
- **Monitoring Points:** ACADA must be able to monitor all hardware-related monitoring points provided by CTA controllable systems up to an overall maximum of 200,000 data points.
- **Data handling:** ACADA must collect event data from all CTA telescope cameras. The data volume the ACADA must handle is: 24 Gb/s for each LST (up to four telescopes in each site), 12 Gb/s for each MST (up to 25 telescopes in CTA-S and 19 in CTA-N), and 2 Gb/s for each SST (up to 70 telescopes in CTA-S).
- **Volume Reduction**: ACADA must be capable of reducing the data volume output by each CTA telescope camera, such that the total data rate delivered to the DPPS at each site is less than 5 Gb/s.
- **ACADA Availability:** The core functionality of ACADA, needed for system control and the safe execution of observations, must be available during 99% of the time in which observations are possible. All ACADA functionality must be available during 95% of the time in which observations are possible.
- **Runtime Environment:** ACADA subsystems and components should be deployable in any standard computing node running a Red Hat Linux derivative in the on-site data centre, and not require specially tailored machines.

ACADA SYSTEM OVERVIEW

Architectural Approach

The ACADA architecture was designed using the Software Platform Embedded System (SPES) methodology [4] implemented via the Unified Modeling Language (UML) and Systems Modeling (SysML) formalisms. Details of our architectural modelling approach are presented in [5]. The ACADA requirements have been analysed and refined via a use case analysis [6]. Functional requirements and use cases have been traced to system functions and components, while quality (non-functional) requirements have been addressed via a set of documented architectural considerations (see section Architectural Considerations).

Context and Main Interfaces

The ACADA system provides the core functionality to support the technical and scientific operations at each CTA site, and therefore, as illustrated in Fig. 1, it contains a significant number of external interfaces.

The SUSS delivers to ACADA the mid-term schedule, which is used by ACADA to determine automatically the night observations, taking into account the weather, incoming transient alerts, and laser traffic control systems on the sites. ACADA delivers information on the proposal execution status to SUSS.

Every morning after the observations, ACADA delivers to DPPS the raw data acquired during the night for further processing, data quality checks, and archiving in the offsite CTA data centres.

ACADA interfaces to individual telescopes (up to 100 in CTA-S and 20 in CTA-N). For each telescope ACADA delivers control commands and configurations. Each CTA telescope, in turn, provides to ACADA monitoring, alarms, logs, and status information. The Cherenkov camera of each CTA telescope provides ACADA camera trigger timestamps, for which the ACADA, in turn, provides confirmation from its central triggering. The camera also provides the raw data to ACADA. In addition to telescopes, ACADA interfaces to the individual devices (e.g. to an individual Lidar), in an analogous way as it does with telescopes.

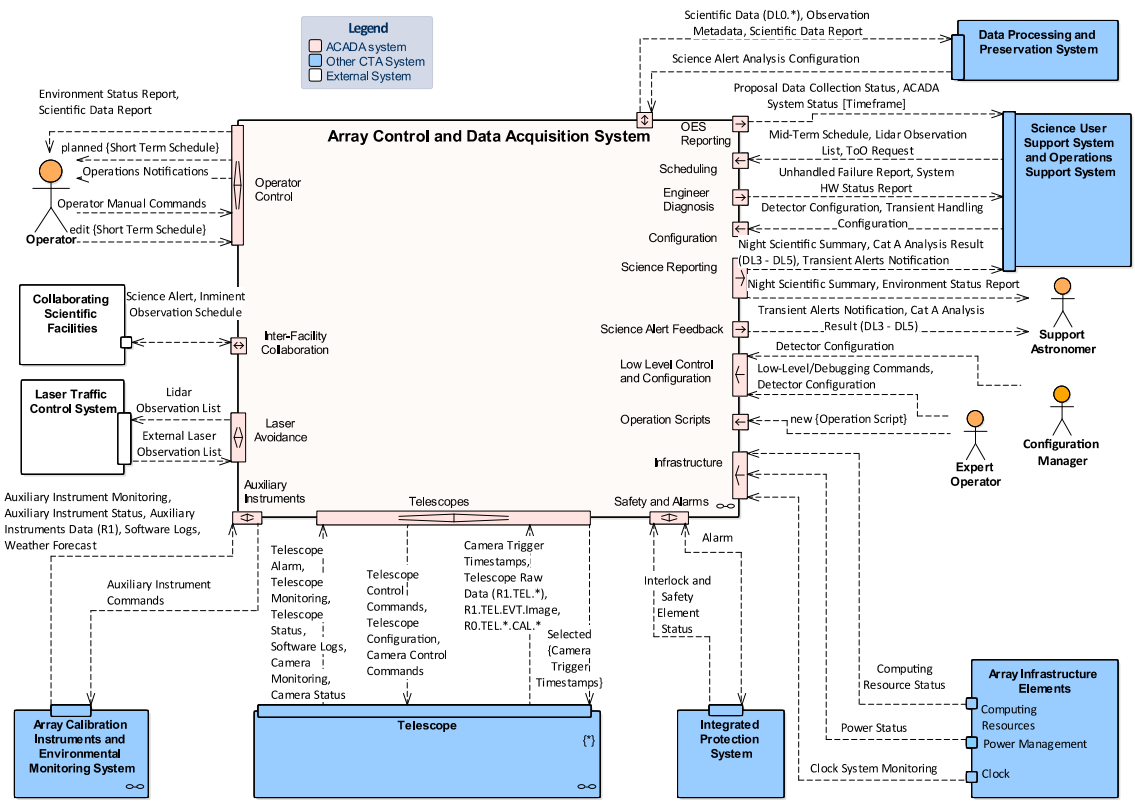

Figure 1: Context view of the ACADA system. The systems external to ACADA are depicted in blue (when part of the CTA System) and white (when external to CTA). The stick figures represent the human actors of the system. See the text for a description of the main interfaces and data flow.

ACADA is also connected with the site infrastructure (e.g. status information about the power management, the clock, and the On-site Data Centre) and the Integrated Protection System (interlocks, alarms).

Finally, ACADA interfaces with other Scientific Facilities for coordination and transient phenomena follow-up scheduling.

System Decomposition

The main components of ACADA are illustrated in Fig. 2 and described in the remainder of this section.

Resource Manager and Central Control (RM&CC) Subsystem responsible for the execution of the scheduling blocks provided by the short-term scheduler. It operates by sending corresponding commands to the telescopes and other controllable array elements. This is done while supervising the ongoing operations, coordinating the allocation of telescopes to sub-arrays, and overseeing the Array Data Handler [7].

Human Machine Interface (HMI) Sub-system that offers a comprehensive view of the status of observations to the operators and support astronomer located in the control room of the CTA installation. It provides the means to interact with the array elements [8].

Array Configuration System (CDB) Sub-systems that stores and distributes the ACADA and supervised systems configuration, including software deployment and the detector configuration settings.

Array Data Handler (ADH) Sub-system responsible for handling the stream of data from the array instrumentation. It includes components to handle the Cherenkov camera data and to reduce the received volume of data. It provides the array-level trigger and handles data acquisition from auxiliary instrumentation [9].

Science Alert Generation Pipeline (SAG) The analysis pipeline running on-line that performs a quick-look analysis of the acquired data. It produces data quality indicators, to be exposed to the support astronomer at the control room and transferred to the DPPS. It also serves to generate internal scientific alerts, submitted to the Transients Handler [10].

Short-Term Scheduler (SCHED) Sub-system responsible for deciding, in real-time, how to group and use the telescopes of a CTA installation to perform nightly operations, based on a mid-term schedule supplied to it before the beginning of nightly operations. This subsystem is also responsible for reacting in real-time to changing environmental conditions and to scientific alert observation requests by the Transients Handler [11].

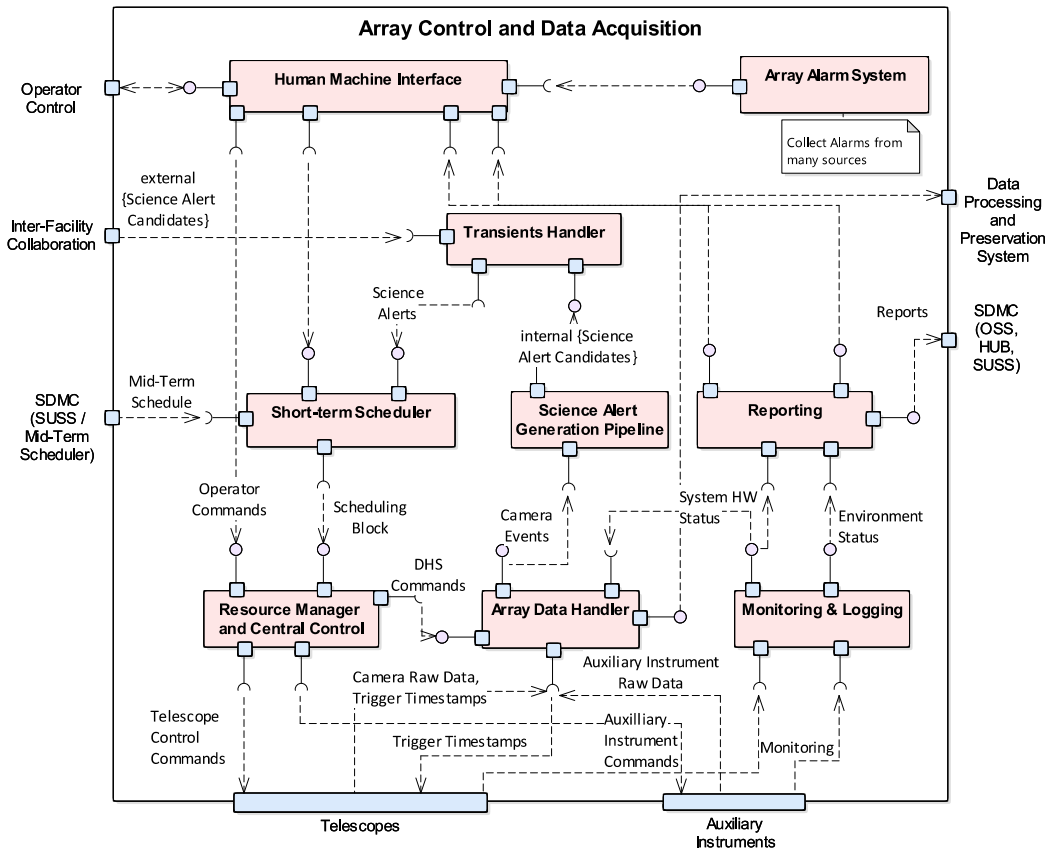

Figure 2: Logical view of ACADA, representing the main components of the system. Only the highest-level components, and the most relevant data elements and interfaces are shown. The diagram uses the UML component notation, and the basic entities are the software components, depicted as pink boxes. Blue squares are "ports" that identify the interfaces of the system. The dashed lines show the flow of data elements.

Transients Handler (TH) The ACADA sub-system responsible for managing internal and external transient science alert candidates by filtering and ranking them, submitting scheduling blocks to SCHED and requesting immediate reaction if needed.

Monitoring and Logging Systems (MON). Components providing services for gathering monitoring data from the telescopes and other devices deployed at the CTA array sites. It is also responsible for making those data immediately available for the operator interface and quick-look quality checking, as well as storing them for offline inspection.

Array Alarm System (AAS) Subsystem that provides the service that gathers, filters, exposes and persists all the relevant alarms raised by the ACADA processes, telescope and auxiliary instrumentation under the supervision of ACADA. It collects alarms from telescopes, auxiliary devices, and ACADA systems.

Reporting System (REP) Sub-system responsible for producing status and quality reports during the night for the HMI and other systems outside ACADA.

ARCHITECTURAL CONSIDERATIONS

ACADA is designed as a highly reliable and fault-tolerant system. It will enable CTA to operate several sub-arrays simultaneously, and to react on timescales of seconds to incoming external and internal science alerts; to coordinate observation between the two array sites; and to promptly detect malfunctioning instruments.

ACADA is by design a distributed system that uses the Alma Common Software (ACS) framework [12] as its basic middleware. ACADA provides its overall functionality via components, where each component is responsible for one or a few clearly defined tasks. The majority of the ACADA components are implemented as ACS components. They run on standard computing nodes of the on-site data centre at the site. The various components interact to provide the functionality required of the ACADA as a whole. ACADA relies on ACS for base functionality towards that interaction, and also uses ACS for component instantiation and other such services.

A selection of the most relevant considerations taken to address the ACADA requirements is presented next.

Component Replacement

Each ACADA component has a supervisor that will monitor its existence and integrity, and orchestrate replacement of that component with its successor as needed. Component supervision and replacement are organized according a to a supervision tree hierarchy [7].

Avoid "Dusty Dark Corners"

Most ACADA functionality should dwell in code that executes regularly. Special-purpose corner-case functionality should be avoided. Rarely invoked error-handling logic, where needed, should be simple.

Shutdown of Unreachable Parts

The operator must not lose control over any part of the array site. However, a network disruption, even a partial one, could disconnect the HMI from parts of the site. What can no longer be controlled by the operator via the HMI, directly or indirectly, must shut down automatically. Hardware devices of all kinds shall be able to realize by themselves when they have become isolated from their supervisor software. In such cases, they have to be able to bring themselves into a "safe state".

Conflicting Orders

Several scenarios could potentially result in conflicting directives, such as a secondary operator using an engineering GUI from a remote location, or the misbehaviour of a component due to a software bug. Such conflicts have to be cleanly resolved. Loosely speaking, in each case, there will be an "old" and "new" command authority. "Old" should lose the conflict and "new" win sole control. The upstream party delegating authority does so by handing out "control tickets". The mere existence of a newer control ticket invalidates all older ones. All commands that change state inside the ACADA command tree must be associated with a ticket.

Prefer Idempotent Operations

Repeated commands arriving at slightly different times, arriving twice, or in an unexpected order may result in unpredictable behaviour. However, due e.g., to timeouts, a client may want to resend a command in order to make sure it arrives. An "idempotent operation" is one that can be applied multiple times without changing the result beyond the initial application. Idempotent operations should be used whenever possible.

Fail Fast with Timeouts

Clients should detect as soon as possible when a service becomes unavailable. Every synchronous or asynchronous service request within ACADA should time out. Services that are not responding within a defined time window will be considered to be in a failed state.

Containing Replacement Impact

The crash and subsequent replacement of an ACADA component should have as little impact as possible on the array operations. A component should either be unlikely to fail, or else, failure and replacement of a component should have little impact. In a detailed analysis, software components of the ACADA system will be categorised into three levels: top, middle and bottom level components, as detailed in the following:

Bottom Components which are required by an ongoing scheduling block. Failure and replacement of such a component come at the price of some loss of scientific value. Correspondingly, during replacement, some observation data will be lost.

Top Central components to the administration of CTA operations as a whole. A component is top-level if it is expected to always exist and be reachable for the proper functioning of the system as a whole. Failure and replacement of such components might not generally result in any observation data loss at all, provided the replacement becomes available quickly enough.

Middle Functionality that is complicated and hence more likely to be buggy and to fail occasionally is best placed in mid-level components. Such components should be designed so they are specifically not centralised, where their failure will only affect a part of the system (e.g., only one scheduling block). Moreover, their failure and replacement will not cause immediate loss of data.

DEVELOPMENT STATUS

Advanced prototypes exist for the main ACADA subsystems. In particular, prototype versions of the ADH [9] and for the SAG [10, 13] are installed at the CTA site in La Palma and are being integrated with the acquisition system of the first LST [14]. Many of the design principles presented in the Architectural Considerations Section have been prototyped and tested in the RM&CC prototype [7].

Furthermore, a small computing cluster composed of 15 machines for supporting the Assembly, Integration, and Verification (AIV) of ACADA [6] had been set up in the DESY data centre in Zeuthen [15]. This environment contains a continuous integration setup based on Jenkins [16] and a software quality assurance environment based on SonarQube [17]. The remaining nodes are available for the overall system and subsystems system integration and performance testing.

CONCLUSION

We have presented the main requirements, architecture and design principles of the CTA ACADA system. The main building blocks of the system and its external interfaces have been outlined. The subsystems within ACADA are being developed and tested with real CTA telescopes. An integration and test environment is in place to support the Assembly, Integration, and Verification of ACADA.

ACKNOWLEDGEMENTS

We gratefully acknowledge financial support from the agencies and organizations listed here: http://www.cta-observatory.org/consortium acknowledgments

REFERENCES

[1] B. Acharya, *et al*., "Introducing the CTA concept", Astroparticle Physics, 43, 3, 2013.
doi:10.1016/j.astropartphys.2013.01.007

[2] I. Oya, *et al.*, "A system architecture approach for the Cherenkov telescope array (Conference Presentation)", in *Proc.* Proc. SPIE 10705, Modeling, Systems Engineering, and Project Management for Astronomy VIII, 107050Y, 2018.
doi: 10.1117/12.2313842

[3] The CTA Consortium, *Science with the Cherenkov Telescope Array*, World Scientific Publishing Co. Pte. Ltd., 2019. doi: 10.1142/10986

[4] K. Pohl, H. Hönninger, R. Achatz and M. Broy (Eds.) *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*, Berlin: Springer, 2012
ISBN: 978-3642346132

[5] I. Oya, *et al.*, "Designing and prototyping the control system for the Cherenkov Telescope Array", in *Proc*. ACAT'17, Seattle, USA, Contribution 42, 2017.
doi: 10.1088/1742-6596/1085/3/032045

[6] E. Antolini, D. Melkumyan, K. Mosshammer, and I. Oya, "Quality Assurance Plan for the SCADA System of the Cherenkov Telescope Array Observatory", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOMPL001, this conference.

[7] D. Melkumyan *et al.*, "Prototyping the Resource Manager and Central Control System for the Cherenkov Telescope Array", presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOPHA092, this conference.

[8] I. Sadeh, I. Oya, D. Dezman, E. Pietriga, and J. Schwarz, "The Graphical User Interface of the Operator of the Cherenkov Telescope Array", in Proc. ICALEPCS'17, Barcelona, Spain, Oct. 2017, pp. 186-191. doi:10.18429/JACoW-ICALEPCS2017-TUBPL06

[9] E. Lyard and R. Walter, "End-to-end data acquisition pipeline for CTA", in Proc. ICRC'17, Busan, Korea, 2017.
doi:10.22323/1.301.0843

[10] A. Bulgarelli, *et al.*, "A prototype for the real-time analysis of the Cherenkov Telescope Array", in *Proc*. SPIE 9145, Ground-based and Airborne Telescopes V, paper 91452X, 2014. doi: 10.1117/12.2054744

[11] J. Colome, *et al*, "Artificial intelligence for the CTA Observatory scheduler", in *Proc* SPIE 9149, Observatory Operations: Strategies, Processes, and Systems V, paper 91490H, 2014. doi: 10.1117/12.2057090

[12] G. Chiozzi, *et al.*, "CORBA-based Common Software for the ALMA project", in *Proc* SPIE 4848, 43, 2002.
doi: 10.1117/12.461036

[13] T. Vuillaume, *et al.*, "hipeCTA: a High Performance Computing library for the CTA data analysis", in *Proc.* ICRC'19, Madison, Wisconsin, USA, 2019.

[14] J. Cortina, *et al.*, "Status of the Large Size Telescopes of the Cherenkov Telescope Array", in *Proc.* ICRC'19, Madison, Wisconsin, USA, 2019.

[15] T. Murach, *et al.*, "Software testing for the CTA observation execution system", in *Proc.* SPIE 10707, Software and Cyberinfrastructure for Astronomy V, paper 107070D, 2018.

[16] Jenkins, https://jenkins.io

[17] Sonarqube, https://www.sonarqube.org