Designing and prototyping the control system for the Cherenkov Telescope Array

I Oya[1], M Füßling[1], P O Antonino[2], M Araya[3], J Borkowski[4], A Bulgarelli[5], J Castroviejo[6], J Colomé[6], V Conforti[5], A Garcia-Piquer[6], J Guárdia[6], L Hagge[7], E Lyard[8], A Morgenstern[2], M Mayer[9], D Melkumyan[1], T Murach[1], L Pizarro[3], I Sadeh[1], T Schmidt[1], U Schwanke[9], G Spengler[9], J Schwarz[10], G Tosti[11], R Walter[8] and P Wegner[1] for the CTA Consortium[12]

[1] DESY, Zeuthen, Germany
[2] Fraunhofer IESE, Kaiserslautern, Germany
[3] UTFSM, Valparaíso, Chile
[4] CAMK, Torun, Poland
[5] I.A.S.F. di Bologna, Italy
[6] IEEC-CSIC, Spain
[7] DESY, Hamburg, Germany
[8] University of Geneva - Departement d'Astronomie, Switzerland
[9] Humboldt-Universitaet zu Berlin, Germany
[10] INAF - Osservatorio Astronomico di Brera, Italy
[11] University of Perugia, Italy
[12] Full consortium author list at http://cta-observatory.org

E-mail: igor.oya.vallejo@desy.de

Abstract. The Cherenkov Telescope Array (CTA) is the next-generation atmospheric Cherenkov gamma-ray observatory. The Observation Execution System (OES) team within the CTA project is designing and prototyping the software to execute the observations and to handle the acquisition of scientific data at GB/s rates. In this contribution we show the OES system as it is being designed using the Unified Modeling Language (UML) and Systems Modeling (SysML) formalisms. In addition, we present the status of the associated prototyping activities.

1. Introduction

The Cherenkov Telescope Array (CTA) [1] will consist of two facilities, one in the southern (Cerro Armazones Chile) and the other in the northern hemisphere (La Palma, Spain). The two sites will contain dozens of telescopes of different sizes, constituting one of the largest astronomical installations under development.

CTA will implement simultaneous automatic operation of multiple sub-arrays of telescopes. It will be capable of quick re-scheduling of observations (within a few seconds), in order to allow observations of elusive transient events. The operation, control, and monitoring of the distributed multi-telescope CTA arrays is inherently complex. As such, they pose new challenges in scientific instrumentation control systems and in particular in the context of ground-based

gamma-ray astronomy. The system providing the functionality to execute the observations and to handle the acquisition of scientific data in CTA is the Observation Execution System (OES).

In the the following we will first introduce the model-driven methodology adopted to design the OES system (Section 2), then present a brief description of the main sub-systems of OES and describe the status of the associated prototypes (Section 3), and finally offer some concluding remarks (Section 4).

2. Model Driven System Design

The OES architecture is designed using the Software Platform Embedded System (SPES) methodology [2] implemented via the Unified Modeling Language (UML) and Systems Modeling (SysML) formalisms.

The architecture is composed of two type of elements: *architecture drivers* and *architecture design*. The architecture drivers specify functional and quality requirements for the system. The architecture design is expressed by different viewpoints. Figure 1 illustrates via an example the main viewpoints we use in the OES architecture, which are the following:

- In the *context* viewpoint the OES system is considered to be a black-box interacting with a set of external systems and *stakeholders* (human actors). The interaction is performed via the exchange of information through the flow of *functional data*; in the example in Fig. 1 we show a system and a stakeholder exchanging information with the OES system.
- In the *functional* viewpoint we derive a network of functions that exchange data with the external entities outside the system as well as between them. The functions are grouped hierarchically, down to the elementary elements named *behavior functions*.
- In the *logical* viewpoint, we create a hierarchical decomposition of the systems, the higher level elements corresponding to the sub-systems contained within the OES (see also Fig. 2). These components are decomposed in a hierarchy of lower level components, down to the elementary implementation units (*behavior components*), that realize the functions previously defined in the functional viewpoint. This viewpoint contains the provided and required interfaces of the components and the concrete (*logical*) data exchanged. In addition to the UML component diagrams which show the static description of the OES system, this viewpoint uses UML activity and sequence diagrams to specify the behavior of the system.
- We describe the deployment specification of the logical components in the *technical* viewpoint (not shown in the example in Fig. 1).

The application of the architecture model serves the following purposes in the teamwork to create the OES system:

- Specification of the architecture design, including interfaces, behavior and data model.
- Relationships (traces) from drivers (functional and quality requirements) to model elements, ensuring consistency.
- Definition of specifications of software to be created by the development teams, which can be either within the CTA organization or external software development companies.
- Vision sharing and scoping (including interface identification and definition), two important ingredients for collaborative software engineering.
- A foundation to implement the project management to build the OES system. The work package product breakdown structure (PBS) and the work breakdown structure (WBS) can be built based on the functional and logical viewpoints. The model can help to identify risks, internal and external dependencies, and to plan staging scenarios in the construction of the OES software.

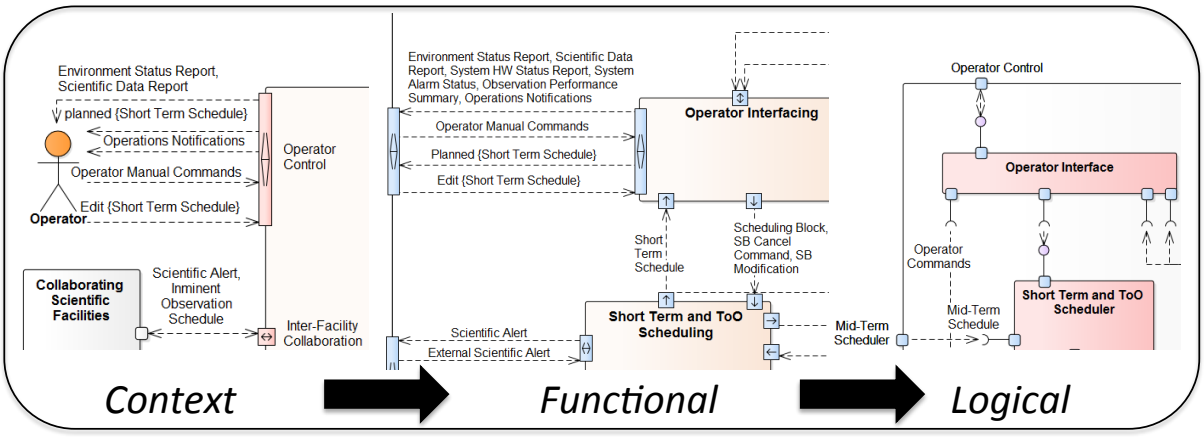

Figure 1. Example to illustrate the main viewpoints of the OES architecture. The arrows exemplify the process of going from an abstract description to a more concrete level of detail, from the Context (left) to the Functional (middle) and then to the Logical viewpoints (right). The diagram utilizes the SysML and UML notation. See text for details.

The OES will be implemented as a distributed software application using the ALMA Common Software (ACS) CORBA-based framework [3, 4]. One instance of the OES will be deployed in each CTA site. Figure 2 shows a diagram indicating the main components of the OES system, described in the next section.

3. OES Subsystems and their Prototypes

The OES design presented here takes as input many concepts from the Array Control and Data Acquisition (ACTL) idea discussed within the CTA Consortium [5, 6]. We have produced advanced prototypes for most OES sub-systems during the past few years, implemented in the ACS framework. The scope of these sub-systems, together with the status of the prototypes is briefly summarized below:

- The **Short Term and ToO Scheduler** is responsible for deciding, at 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 the night operations. This sub-system is also responsible for reacting at real time to changing environmental conditions and external or internal scientific alerts (scientific alerts are also known as ToOs; target of opportunity alerts). The scheduler prototype is implemented in C++, and uses advanced artificial intelligence algorithms (meta-heuristic optimization) [10, 11].
- The **Manager and Central Control** is responsible for the execution of the scheduling blocks provided by the scheduler by sending corresponding commands to the telescopes, while supervising the ongoing operations, bookkeeping the allocation of telescopes to sub-arrays, and overseeing the Data Handling System. A prototype of this sub-systems exists, implemented in Java and using a supervision tree architecture. The prototype includes a dedicated Python environment to run operations scripts. The latter are the high level procedures implementing the actions to be performed by the telescopes in a sub-array during a particular operation.
- The **Data Handling System** is responsible for getting the stream of data from the Cherenkov cameras of the CTA telescopes and handling these data, which arrive at a rate of the order of GB/s, according to the sub-array they participate in. As such, it contains the central trigger, data acquisition, local storage, and a pipeline to provide scientific alerts

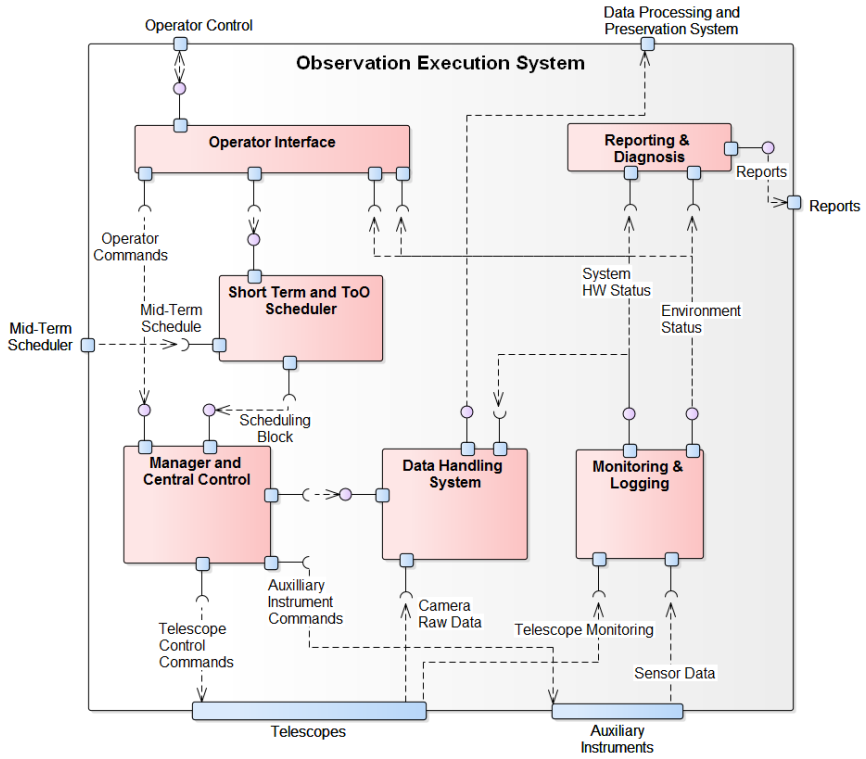

Figure 2. Logical view of OES, representing the main components of the system. Only the highest level components, and the most relevant data elements and interfaces are shown. The diagram uses the UML component notation, and the basic entities are the software components, depicted as pink boxes. The dashed lines show the flow of data elements and the arrows the direction and sign of the flow.

at real time. More details on the existing prototype of the data acquisition, implemented in C/C++, is provided in [7]. The concept and design ideas of the science alerts pipeline is presented in [8].

- The **Operator Interface** offers to the operator located in the control room of the CTA installation a comprehensive view of the status of the observations and the hardware and means to interact with the system. Our prototype [9] is based on web technologies, with the front-end implemented in JavaScript and the back-end in Python (Pyramid Web Framework). Figure 3 shows a few examples of existing prototype panels.
- The **Monitoring System** is responsible for monitoring data items from the telescope and other devices deployed at the CTA sites, and making those data available for the operator interface, for quick-look quality checking, as well as for detailed inspection. In our estimates, we assume a total of 150,000 data items to be monitored at a rate of about 1 Hz. Based on the assumed number of data items, we are prototyping solutions that are using redis, MongoDB and Cassandra nosql data base technologies.
- The **Reporting and Diagnosis** sub-system is responsible for gathering the relevant data from the other OES subsystems in order to produce status and quality reports during the night for the Operator Interface and for other systems outside the OES. This system has not yet been prototyped.

A preliminary integration of the OES sub-system prototypes mentioned above is underway, starting with integration of the Central Control with the Short Term Scheduler and the Data

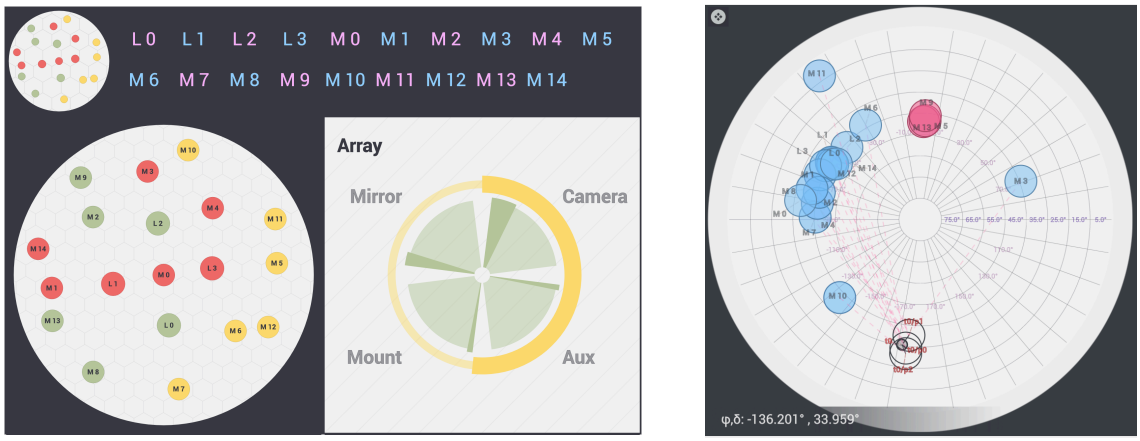

Figure 3. Two panels from the operator User Interface prototype. The *array zoomer* (left panel) shows the status of array elements at different levels of detail by implementing multi-scale navigation with semantic zooming concepts (see [9] for details). The figure in the right panel shows the position of telescopes (blue/red circles) on the sky in polar coordinates.

Handling System.

4. Conclusions

We have set up a model-driven approach to design the OES software. The system has been decomposed in several sub-systems, most of them already having associated prototypes. The pre-integration of the sub-system prototypes is the first step in the construction phase of the OES system. Our experience during the last couple of years is that the usage of the formal architecture approach presented here improves the communication of expectations and interface definitions between the different development teams, which are physically distributed around the world. Following this, we conclude that such model-driven approach will be instrumental for the success of the OES implementation in the future.

Acknowledgments

The authors acknowledge the work of the ACTL team members of the CTA consortium during the last years, which was instrumental for the definition of the OES system. We gratefully acknowledge financial support from the agencies and organizations listed here: http://www.cta-observatory.org/consortium_acknowledgments

References

[1] Acharya B, Actis M, Aghajani T, et al. 2013 *Astroparticle Physics* **43**, 3
[2] Pohl K, Hönninger H, Achatz R, Broy M (Eds.) 2012 *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology* (Berlin: Springer)
[3] Chiozzi G, Jeram B, Sommer H, et al. 2014 *Proc. SPIE* 9149, 91490H
[4] Schwarz J, Sommer H and Farris A 2004 *ASP Conference Series* 314, 634
[5] Oya I, Füßling M, Antonino PO, et al 2016 *Proc. SPIE* 9913, 991303
[6] Füßling M, Oya I, Balzer A, et al 2016 *Proc. SPIE* 9913 99133C
[7] Lyard E, Walter R 2017 *End-to-end data acquisition pipeline for CTA*, *Proc. ICRC* (*Preprint* arXiv:1709.04203)
[8] Bulgarelli A, Fioretti V, Zoli A, et al 2014 *Proc. SPIE* 9145 91452X
[9] Sadeh I, Oya I, Schwarz J and Pietriga E 2016 *Proc. SPIE* 9913, 99130X
[10] Colomé J, Colomer P, Campreciós J, et al 2012, *Proc. SPIE* 5496, 205-218
[11] Garcia-Piquer A, Morales JC, Ribas I, et al 2017 *Astronomy & Astropysics* **604**, A87.