Software design of the ASTRI camera server proposed for the Cherenkov Telescope Array

Conforti, Vito, Trifoglio, Massimo, Gianotti, Fulvio, Malaguti, Giuseppe, Bulgarelli, Andrea, et al.

Vito Conforti, Massimo Trifoglio, Fulvio Gianotti, Giuseppe Malaguti, Andrea Bulgarelli, Valentina Fioretti, Andrea Zoli, Osvaldo Catalano, Milvia Capalbi, Pierluca Sangiorgi, "Software design of the ASTRI camera server proposed for the Cherenkov Telescope Array," Proc. SPIE 9913, Software and Cyberinfrastructure for Astronomy IV, 99133G (26 July 2016); doi: 10.1117/12.2230016

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2016, Edinburgh, United Kingdom

Software design of the ASTRI camera server proposed for the Cherenkov Telescope Array

Vito Conforti [*a], Massimo Trifoglio [a], Fulvio Gianotti [a], Giuseppe Malaguti [a], Andrea Bulgarelli [a], Valentina Fioretti [a], Andrea Zoli [a], Osvaldo Catalano [b], Milvia Capalbi [b], Pierluca Sangiorgi [b], for the ASTRI Collaboration[c] and the CTA Consortium[d]

[a] INAF, IASF-Bologna, Via Pietro Gobetti 101, 40129 Bologna, Italy
[b] INAF, IASF-Palermo, Via Ugo La Malfa 153, 90146 Palermo, Italy
[c] http://www.brera.inaf.it/astri
[d] http://www.cta-observatory.org

ABSTRACT

The Italian National Institute for Astrophysics (INAF) is leading the ASTRI project within the ambitious Cherenkov Telescope Array (CTA), the next generation of ground-based observatories for very high energy gamma-ray astronomy. In the framework of the small sized telescopes (SST), a first goal of the ASTRI project is the realization of an end-to-end prototype in dual-mirror configuration (2M) with the camera composed of a matrix of Silicon photo-multiplier sensors managed by innovative front-end and back-end electronics. The prototype, named ASTRI SST-2M, is installed in Italy at the INAF "M.G. Fracastoro" observing station located at Serra La Nave, 1735 m a.s.l. on Mount Etna, Sicily. As a second step, the ASTRI project is focussed on the implementation of a mini-array composed at least of nine ASTRI telescopes and proposed to be placed at the CTA southern site. This paper outlines the design of the camera server software that will be installed on the ASTRI mini-array. The software is based on the version installed on the ASTRI SST-2M prototype operating in a single telescope configuration. The migration from single telescope to mini-array context has required additional interfaces in order to guarantee high interoperability with other software and hardware components. In the mini-array configuration each camera communicates with its own camera server via a dedicated high rate data link. The primary goal of the camera server is to acquire the bulk data, packet by packet, without any data loss and to timestamp each packet very precisely. During array operation, the camera server receives from the SoftWare Array Trigger (SWAT) the list of science events that participate in stereo triggered events. These science events, and all others that are flagged either by the camera as interleaved calibration or by the camera server as possible single-muon events, are sent to the Array DAQ. All remaining science events will be discarded. A suitable buffer is provided to perform this processing on all the incoming event packets. The camera server provides interfaces to the array control software to allow for monitoring and control during array operations. In this paper we present the design of the camera server software with particular emphasis on the external interfaces. In addition we report the results of the first integration activities and performance tests.

Keywords: Cherenkov Telescope Array, CTA, software design, ASTRI, camera server

1. INTRODUCTION

The Italian National Institute for Astrophysics (INAF) is leading the ASTRI project[1] proposed for the ambitious Cherenkov Telescope Array (CTA)[2], the next generation of ground-based observatories for very high energy gamma-ray astronomy. In the framework of the small sized telescopes[3], a first goal of the ASTRI project is the realization of an end-to-end prototype in a dual-mirror configuration (SST-2M) with the camera at the focal plane composed of a matrix of Silicon photo-multiplier sensors managed by an innovative front-end and back-end electronics[4]. The ASTRI SST-2M prototype is installed in Italy at the INAF "M.G. Fracastoro" observing station located at Serra La Nave, 1735 m a.s.l. on Mount Etna, Sicily. As a second step, the ASTRI project is focussed on the implementation of a mini-array[5] composed at least of nine ASTRI telescopes and proposed to be placed at the CTA southern site. A first version of the software to be installed on the camera server of the ASTRI SST-2M prototype is being deployed and tested[6]. It is aimed at the

[*] conforti@iasfbo.inaf.it, http://www.iasfbo.inaf.it/~conforti/

Software and Cyberinfrastructure for Astronomy IV, edited by Gianluca Chiozzi, Juan C. Guzman, Proc. of SPIE Vol. 9913, 99133G · © 2016 SPIE
CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2230016

acquisition, storage and display of the camera data. This version is being re-engineered for the ASTRI telescopes in mini-array configuration, where each camera server, one per telescope, has to interface with the array control and data acquisition (ACTL) system[7]. In this configuration the camera server provides a different management of the event time stamping and forwards the stereo triggered events to the array data acquisition (DAQ) pipeline. In this paper we present the preliminary design of the camera server software envisaged for the ASTRI mini-array[8]. The design is a conceptualization of a subject (the system or software under design) that embodies its essential characteristic, demonstrates a means to fulfil its requirements, serves as a basis for analysis and evaluation and can be used to guide its implementation. The key life cycle product that drives a software design is typically the requirements specification which drives the design and design constraints to be considered. After the introduction of the camera server software specification requirements in the next section, we present, in section 3, the preliminary design of the camera servers for the ASTRI mini-array telescopes.

2. THE CAMERA SERVER REQUIREMENT SPECIFICATIONS

The Software Requirements Specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements. The SRS establishes the basis for an agreement between customer and supplier on what the software product is to do as well as what it is not expected to do. The software requirements specification permits a rigorous assessment of requirements before the design can begin and reduces later redesign. The camera server software specifications implement the ASTRI requirements which have been collected under the CTA requirements. We present in the next sub-sections the functional requirements through the Unified Modeling Language (UML) use cases, and the non-functional requirements in text-form.

2.1 Functional requirements specification

During operations the ASTRI camera servers, one for each ASTRI telescope, have to perform data stream acquisition, buffering, preliminary data reduction and forwarding to the array DAQ. The use case technique captures the behaviour of a system as it appears to an outside user. It partitions the system functionality into transactions meaningful to the actors. The UML diagram in Figure 1 depicts the camera server use cases.

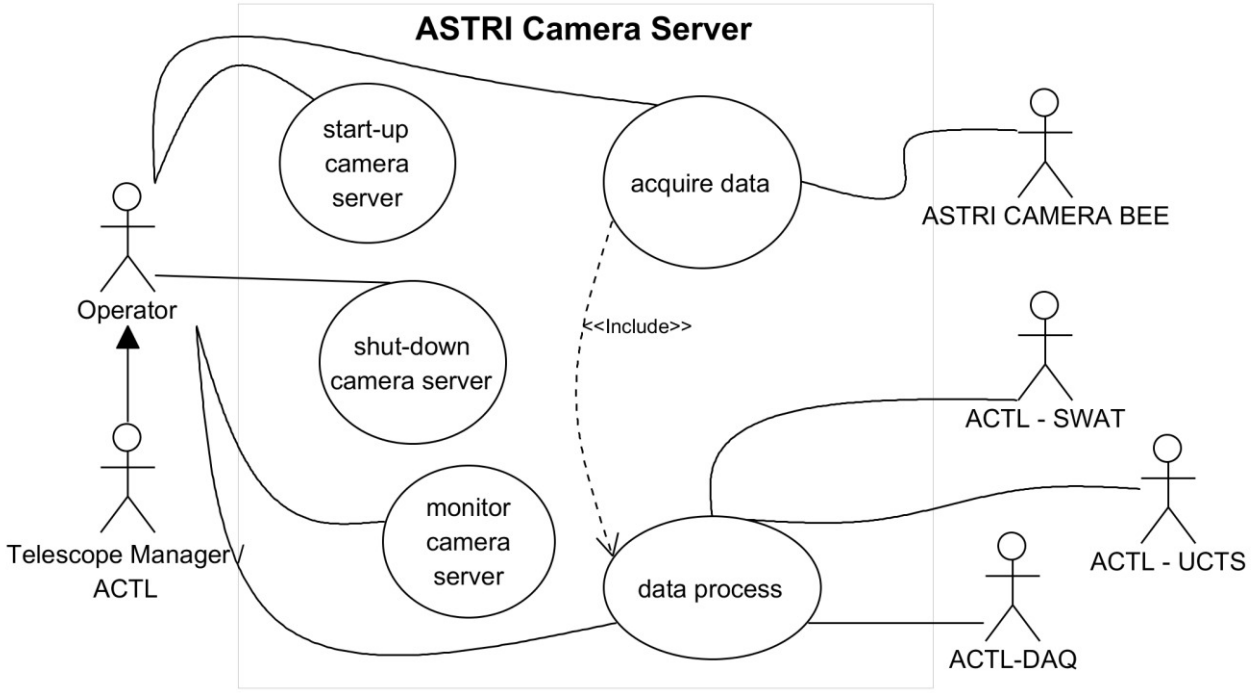

Figure 1. UML ASTRI camera server use case diagram.

The actors are:

- Operator (Telescope Manager): this is either the engineer during the maintenance activities, or the operator who has to perform the night observations with the ASTRI telescope;
- ASTRI camera BEE: the ASTRI camera Back End Electronics;
- SWAT (Softare Array Trigger): the system responsible for detecting the stereo trigger events;
- CDTS (Clock Distribution Time System): ensures the high precision synchronization among the telescope cameras;
- DAQ (Data AcQuisition): acquires data from all the camera servers.

The following tables detail, through the most significant information, the use case presented in the UML diagram (Figure 1).

Table 1. Start-up camera server – use case detail.

Title	**Start-up camera server**
Actors	Operator (Telescope Manager)
Pre-conditions	The camera server is off
Post-conditions	The camera server is ready for operations
Main success scenario	1. The operator commands the camera server to start 2. The camera server starts 3. The camera server software notifies the operator that it is ready for operations
Extensions	2a. The camera server does not start 2a.1 The camera server provides an error message

Table 2. Shut-down camera server – use case detail.

Title	**Shut-down camera server**
Actors	Operator (Telescope Manager)
Pre-conditions	The camera server is ready for operations
Post-conditions	The camera server is off
Main success scenario	1. The operator commands the camera server to shut-down 2. The camera server shuts down 3. The camera server software notifies the operator its off status
Extensions	2a. The camera server does not shut-down 2a.1 The camera server provides the error message

Table 3. Monitor camera server – use case detail.

Title	**Monitor camera server**
Actors	Operator (Telescope Manager)
Pre-conditions	The camera server is ready for operations
Post-conditions	The camera server is ready for operations and it is monitored
Main success scenario	1. The camera server continuously sends its status to the operator
Extensions	Not provided

Table 4. Acquire data – use case detail.

Title	**Acquire data**
Actors	ASTRI camera BEE, Telescope Manager
Pre-conditions	The camera server is ready for operations
Post-conditions	The camera server is acquiring data
Main success scenario	1. The ASTRI camera BEE continuously sends data to the camera server packet by packets 2. The camera server receives data packets 3. The camera server processes data packets
Extensions	2a The camera server does not receive data packets 2a.1 The camera server notifies the error to the operator

Table 5. Process data – use case detail.

Title	**Process data**
Actors	SWAT, CDTS, DAQ, Telescope Manager
Pre-conditions	The ASTRI camera server is acquiring data packet
Post-conditions	The ASTRI camera server has processed data packet
Main success scenario	For each data packet acquired by the camera server: 1. The camera server buffers the data packet 2. If the packet contains either calibration or variance data a. The camera server sends data to the DAQ

	3. If the packet contains scientific data a. The CDTS sends the event timestamps to the camera server b. The camera server checks the timestamp event with those received from the CDTS c. The SWAT sends the stereo triggered event list to the camera server d. If the packet contains a stereo triggered event then the camera sends the packet to the DAQ
Extensions	1a The camera server has lost the packet 1a.1 The camera server notifies the operator the packet loss 2a The camera server does not send packet to the DAQ 2a.1 The camera server notifies the operator the sending error 3b The camera server timing check fails 3b.1 The camera server notifies the operator that the timing check is failed 3d The event packet is not a stereo trigger event 3d.1 The camera server applies the muon detection algorithm to the event packet 3d.2 If the event is a potential muon event then the camera server sends the packet to the DAQ, else the packet is discarded

2.2 Non-functional requirements specification

During the operations the ASTRI camera servers, one for each ASTRI telescope, shall be verified according to the following non-functional requirements:

- The camera server shall be able to acquire at least 600 packets per second;
- The camera server shall acquire the 99,9% of data packets created by the ASTRI camera BEE each night.

3. THE CAMERA SERVER DESIGN

This section presents the camera server design envisaged for the ASTRI telescopes of the mini-array. The software design plays a key role in the development of software products: it is a phase of the complete software engineering life cycle. The fundamental software engineering life cycle phases include requirements, design, construction, test, and maintenance. The final implementation phase begins once the design has been executed and the requirement specifications can be traced to a section of the software design models. The camera server software will be deployed in a physical server for each ASTRI camera. In this section we describe the camera server software through the viewpoints[9].

3.1 Context viewpoint

This viewpoint represents the services provided by a design subject with reference to an explicit context. Figure 2 depicts the context diagram which defines the boundary between the system and its environment, showing the entities that interact with it.

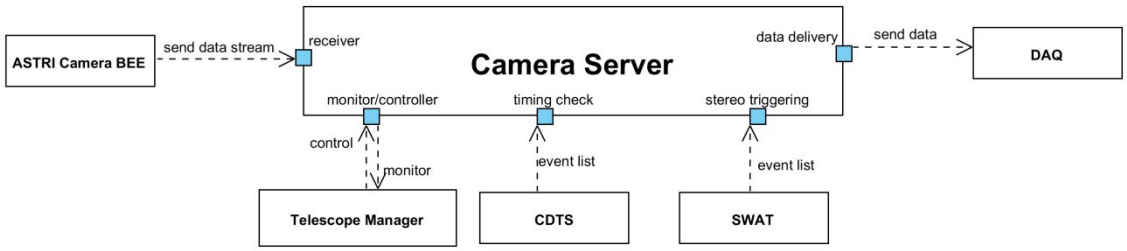

Figure 2. ASTRI camera server context diagram.

The design entities which interact with the camera server are:

- The ASTRI camera BEE (Back End Electronics) which sends the camera data stream packet by packet through the dedicated TCP/IP socket connection. The data packet format is defined through the ESA telemetry standard[10].
- The Telescope Manager is devoted to monitoring and controlling all the telescope components.
- The CDTS (Clock Distribution Time System) synchronizes all the camera clocks. The CDTS provides the trigger time to the camera server for every camera trigger signal. The camera server receives the event list and checks that the time received from the CDTS is the same time set from the camera BEE.
- The SWAT inspects the telescope event time-stamps, and sends the stereoscopic events list to the involved camera servers. The camera server prepares and delivers to the DAQ either the stereoscopic or mono-muon events.
- The DAQ receives the muon-events, stereoscopic events, and calibration and variance data.

3.2 Composition viewpoint

The composition viewpoint, depicted in Figure 3, describes the way the design subject is structured into constituent parts and establishes the roles of those parts. This view identifies the major design constituents of the design object in order to localize functionalities.

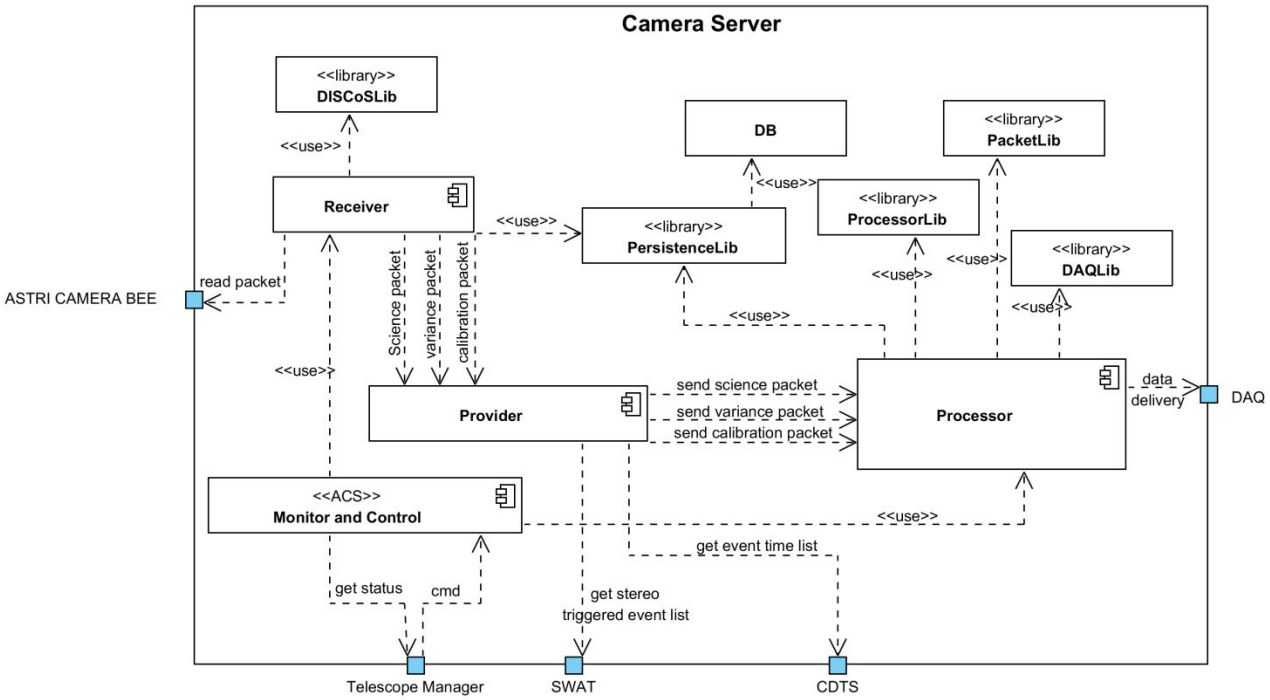

Figure 3. ASTRI camera server UML component diagram.

The software design of the mini-array camera servers is a refurbishment of the camera server software design developed for the ASTRI SST-2M prototype[6], in order to reuse the most software components. The *Receiver* component is in charge of the data acquisition from the ASTRI camera BEE packet by packet through a TCP/IP connection. In addition, once the packet is acquired, the receiver performs the preliminary data packet transmission check. If this check is successful, then the *Receiver* queues the packet in a specific circular buffer depending from its own packet type. The ASTRI camera provides three main packet types: scientific, calibration, variance. The *Receiver* uses the *DISCoSLib*[14]: a software library aimed at the data stream managing. The *DISCoSLib* has been used also to support the ground test activities in the space projects. The *Provider* is responsible for the buffered packet management. In particular it ensures

that the Processor component takes in charge all the queued packets within a predefined time window. The *Provider* gets, from the *CDTS*, the camera trigger high precision timestamp list to apply to the science data packets. Also the provider gets, from the *SWAT*, the stereo trigger event list to tag the science packets, and eventually apply the stereo trigger selection. The *Processors* component reads the data packets from the provider buffers and performs the preliminary data processing. The *Processors* component is implemented in one processor for packet type. Each processor is synchronized with the relative queue buffer: it processes the data, packet by packet, once a packet is available in the buffer. The processors for variance and calibration data, decode and prepare the packet, for the DAQ delivery, once it is available in the buffer queue. The science processor instead, once the packet is available in the buffer queue, decodes the packet and performs the event muon-recognition[11]. The science processor delivers to the DAQ either the muon tag event or the stereo trigger tag event. The remaining science data are discarded. The processors use the *ProcessorLib* to synchronize the buffers of the *Provider*, and the *PacketLib*[13] to decode the packets. Also the *PacketLib* and *ProcessorLib* have been used to support the ground test activities in the space projects. The processors exploit also the *DAQLib* methods to prepare and delivery the data to the DAQ. The *PersistenceLib* exploits the MySql server database to support the configuration and operations of the *Receiver* and the *Processor* components. The *Monitor and Control* component is built on the ACS[15] (Alma Common Software) framework; it is in charge of monitoring and controlling of the whole camera server software. *Monitor and Control* interfaces with the other telescope ACS components[12]. The ACS provides the IDL (Interface Definition Language) to implement synchronous and asynchronous remote calls, and the notification channel to implement event-driven programming through the publish/subscribe mechanism.

3.3 Interaction viewpoint

The interaction viewpoint defines strategies for interaction among entities. This interaction is part of dynamic behaviour of the system. We focus here on the interactions among *Receiver*, *Provider* and *Science processor* when the camera server receives a science data packet. The UML sequence diagram in Figure 4 details these interactions.

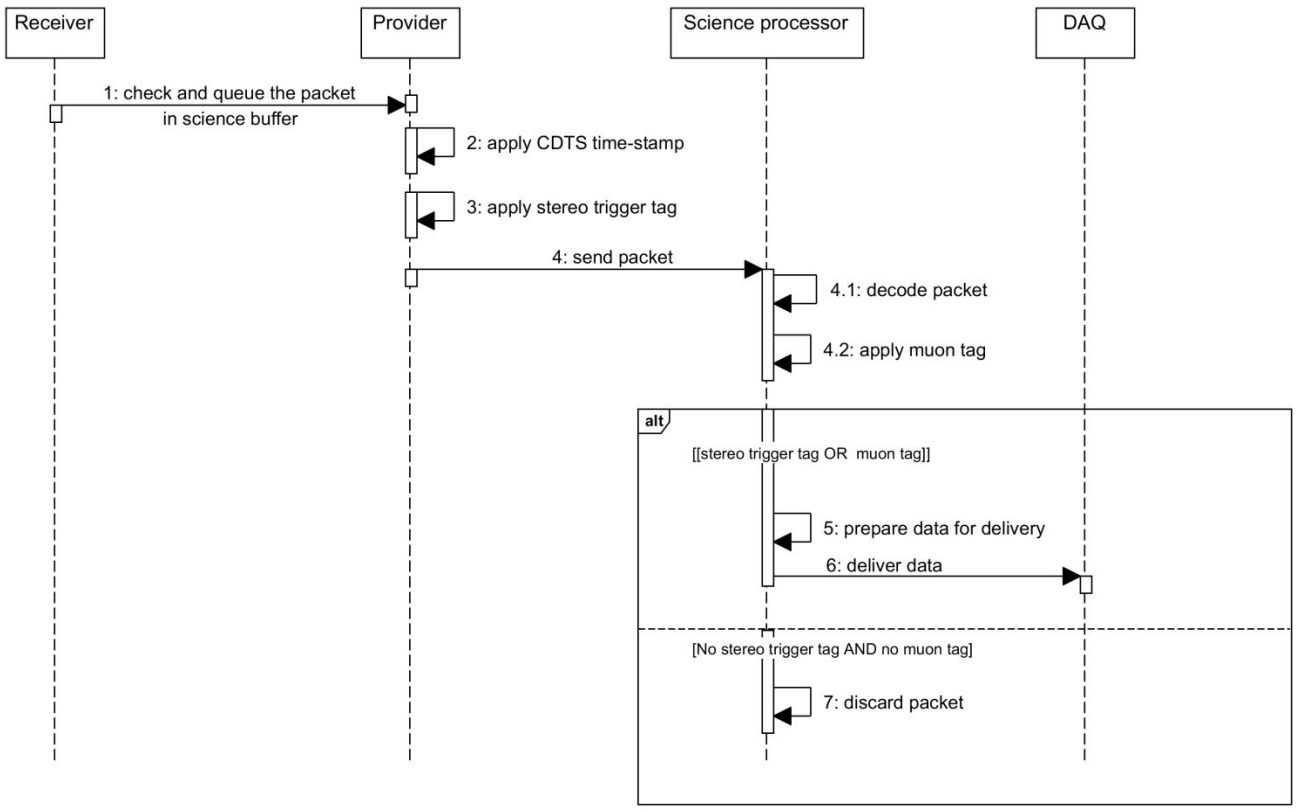

Figure 4. ASTRI camera server science data management UML sequence diagram.

The scenario in Figure 4 depicts the following steps:

1. once the receiver acquires the science packet it performs a preliminary check and queues the packet in the science buffer;
2. the provider awaits the CDTS time-stamp list, then looks for the related packet time-stamp in that list, and sets it into the packet;
3. the provider awaits the SWAT event list, then looks for the related event packet in that list and sets the stereo trigger tag;
4. the provider sends the packet to the science processor which decode the packet and performs the muon-recognition;
5. if the processor finds that the event data is either a stereo-trigger event or a muon event then it prepares the data for delivery;
6. the processor delivers the data;
7. else the processor finds that the event is not either a stereo-trigger event or a muon event and it is discarded.

4. CONCLUSIONS

We presented the ASTRI camera server software design, in section 3, according to the requirement specification detailed in section 2. We have designed the camera server for the ASTRI telescopes under a mini-array configuration in order to extensively re-use the software components which have already been implemented and tested for the ASTRI SST-2M prototype, acting as single telescope.

ACKNOWLEDGMENTS

This work is supported by the Italian Ministry of Education, University, and Research (MIUR) with funds specifically assigned to the Italian National Institute of Astrophysics (INAF) for the Cherenkov Telescope Array (CTA), and by the Italian Ministry of Economic Development (MISE) within the "Astronomia Industriale" program. We acknowledge support from the Brazilian Funding Agency FAPESP (Grant 2013/10559-5) and from the South African Department of Science and Technology through Funding Agreement 0227/2014 for the South African Gamma-Ray Astronomy Programme. We gratefully acknowledge support from the agencies and organizations listed under Funding Agencies at this website: http://www.cta-observatory.org/.

This paper has gone through internal review by the CTA Consortium.

REFERENCES

[1] Pareschi, G, et al, for the CTA Consortium, "The dual-mirror Small Size Telescope for the Cherenkov Telescope Array", Proc.33rd ICRC, arXiv:1307.4962 (2013).
[2] Acharya, B. S., et al., The CTA Consortium, "Introducing the CTA concept", Astroparticle Physics 43, 3-18 (2013).
[3] Montaruli, T., Pareschi, G., Greenshaw Tim "The small size telescope projects for the Cherenkov Telescope Array", Proc. 34th ICRC, arXiv:1508.06472 (2015).
[4] Catalano, O., Maccarone, M.C., et al, for the ASTRI Collaboration and the CTA Consortium, "The camera of the ASTRI SST-2M prototype for the Cherenkov Telescope Array", Proc. SPIE 9147, doi: 10.1117/12.2055099, (2014).
[5] Vercellone, S., for the ASTRI Collaboration and the CTA Consortium, "The ASTRI mini-array within the Cherenkov Telescope Array", Proc. RICAP 2014, EPJ-EDP Sciences, arXiv: 1508.00799, (2015).
[6] Conforti, V., Trifoglio, et al, for the ASTRI Collaboration and the CTA Consortium, "The ASTRI SST-2M telescope prototype for the Cherenkov Telescope Array: camera DAQ software architecture", Proc. SPIE 9152, doi:10.1117/12.2054519 (2014).

[7] Oya, I., Fuessling, M., Antonino, P., Conforti, V., Hagge, L., Morgenstern, A., Tosti, G., Schwanke, U., Schwarz, J., Wegner, P., Colome, P., Lyard, E., for the CTA Consortium, "The software architecture to control the Cherenkov Telescope Array", *these proceedings* (2016).
[8] Tosti, G., Schwarz, et al, , for the ASTRI Collaboration and the CTA Consortium, "The ASTRI MASS Software System", Proc. SPIE 9152, doi: 10.1117/12.2055067, (2014).
[9] "IEEE Standard for Information Technology – Systems design – Software Design descriptions" – IEEE computer Society, IEEE Std 1016TM-2009 Revision, 20/07/2009]
[10] "Packet Telemetry Standard, ESA-PSS-04-106" Issue 1 – European Space Agency, (1988)
[11] Maccarone, M.C., Mineo, T., Capalbi M., Conforti, V., "Pre-selecting muon events in the camera server of the ASTRI telescopes for the Cherenkov Telescope Array", *these proceedings* (2016).
[12] Conforti, V., Tosti, G., Schwarz, J., et al, , for the ASTRI Collaboration and the CTA Consortium, "The high level interface defintions in the ASTRI/CTA Mini Array Software System (MASS)" Proc. 24th ADASS (2014).
[13] Bulgarelli, A., Gianotti, F., Trifoglio, M. "PacketLib: A C++ Library for Scientific Satellite Telemetry Applications" Proc. 12th ADASS (2003).
[14] Gianotti, F., Trifoglio, M., "DISCoS Detector-Independent Software for On-Ground Testing and Calibration of Scientific Payloads Using the ESA Packet Telemetry and Telecommand Standards" Proc. 10th ADASS (2001).
[15] Chiozzi, G., Caproni, A., Jeram, B., Sommer, H., Wang, V., Plesko, M., Sekoronaja, M., Zagar, K., Fugate, D.W., Harrington, S., Di Marcantonio, P., Cirami, R. "Application development using the ALMA common software", Proc. SPIE 6724, doi: 10.1117/12.671707, (2006).