ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA004

PROCEDURES OF SOFTWARE INTEGRATION TEST AND RELEASE FOR ASTRI SST-2M PROTOTYPE PROPOSED FOR THE CHERENKOV TELESCOPE ARRAY

V. Conforti*,A. Bulgarelli, V. Fioretti, F. Gianotti, G. Malaguti, M. Trifoglio,
INAF – IASF, Bologna, Italy
M. Bartolini, A.Orlati, INAF – IRA, Bologna, Italy
O. Catalano, P. Sangiorgi, INAF – IASF, Palermo, Italy
L.A. Antonelli, S. Gallozzi, S. Lombardi, F. Lucarelli, M. Mastropietro, V. Testa,
INAF – O.A. Roma, Monteporzio Catone, Italy
F. Russo, INAF – O.A., Pino Torinese (Torino), Italy
P. Bruno, A. Costa, A. Grillo, F. Vitello, INAF – O.A., Catania, Italy
R. Canestrari, J. Schwarz, S. Scuderi, S. Vercellone, INAF – O.A. Brera, Merate, Italy
E. Antolini, G. Tosti, Università degli studi di Perugia, Perugia, Italy
for the CTA ASTRI Project

Abstract

The Cherenkov Telescope Array (CTA) project is an international initiative to build a next generation ground-based observatory for very high energy gamma-rays. Three classes of telescopes with different mirror size will be located in the northern and southern hemispheres. The ASTRI mini-array of CTA preproduction is one of the small sized telescopes mini-arrays proposed to be installed at the CTA southern site. The ASTRI mini-array will consist of nine units based on the end-to-end ASTRI SST-2M prototype already installed on Mt. Etna (Italy). The mini-array software system (MASS) supports the end to end ASTRI SST-2M prototype and mini-array operations. The ASTRI software integration team defined the procedures to perform effectively the integration test and release activities. The developer has to properly use the repository tree and branches according to the development status. We require that the software includes also specific sections for automated tests and that the software is well tested (in simulated and real system) before any release. Here we present the method adopted to release the first MASS version to support the ASTRI SST-2M prototype test and operation activities.

INTRODUCTION

The Italian National Institute for Astrophysics (INAF) is leading the ASTRI project [1] proposed for the ambitious Cherenkov Telescope Array (CTA) [2]. In the framework of the small size class of telescopes, a first step of the ASTRI project is the realization of an end-to-end prototype in a dual mirror configuration (SST-2M) [3] with the camera at the focal plane composed of a matrix of silicon photo sensors managed by 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, 1753 m a.s.l. on Mount Etna, Sicily [5]. As a second step, as part of the early CTA southern site, the project includes the implementation of the ASTRI mini-array [6] composed of nine ASTRI telescopes. The ASTRI SST-2M prototype has been earlier verified and tested with the engineering software released in beta version. Later on we defined a software integration method to support the integration test and software release activities for the next software version [7]. We are adopting an iterative incremental approach for the development of the software, then we foresee many other software releases before to publish a stable software version which fulfils the whole ASTRI system requirements. In addition we plan to use this method also during the maintenance activities such as debugging and implementation of new functionalities. Figure 1 depicts the whole ASTRI software [8]. The Archive & Data Analysis System runs off site. It provides the management of permanent archive, the data analysis, the proposal management and the tools to access data and proposal [9]. The Observatory Control System (OCS) provides the graphic user interface (also for engineering purposes) for the management of the whole on site system. The resource manager is responsible to start up and monitor the resources in order to minimize any resource fault. The Logger stores continuously the system status; the OCS MASTER monitors and controls the operations. The Observatory Control System uses the on site repository through the TMCDB (the database for the software configurations and the monitor points), the Data Capturer (to support the on site analysis) and the DAQ (Data AcQuisition system) which archives the science data received from the camera server. The Device Control, used also by the Observatory Control System, has in charge the telescope and the ICT (Information and Communication Technologies). This software (written in high level programming such as C++, Java or Python) is built within the ACS (Alma Common Software) framework [10]. The Device Control components interact with the device firmware through OPC-UA [11] and with other ACS component through the ACS services. The camera server which deploys the DAQ software acquires the bulk data from the

* conforti@iasfbo.inaf.it

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA004

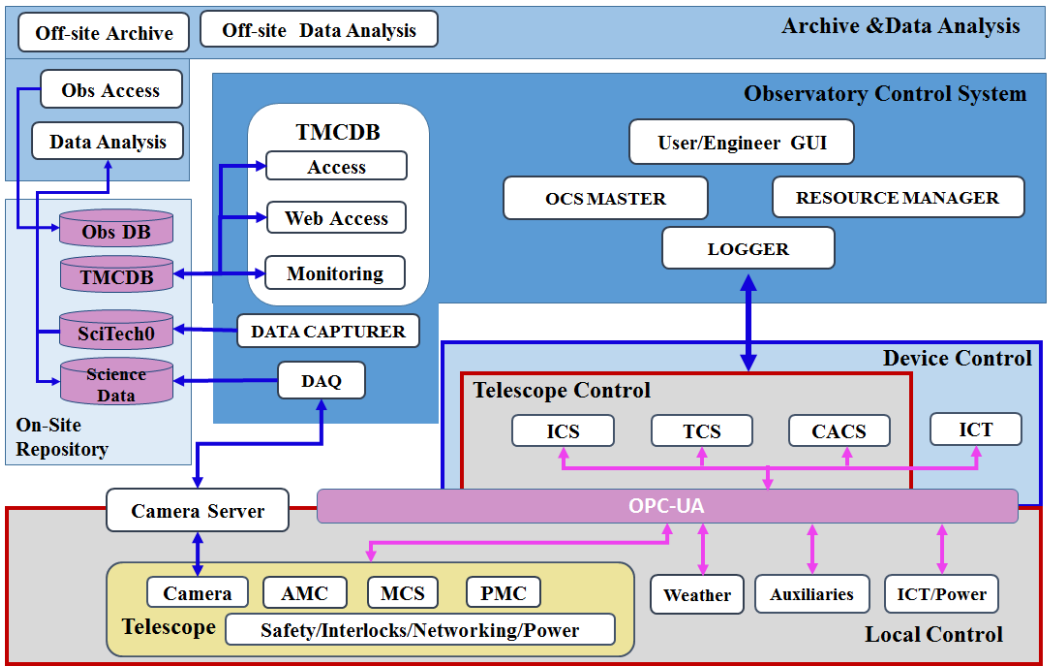

Figure 1: ASTRI MASS software architecture.

Camera BEE (Back End Electronic) and performs the pre-processing [12]. The ICS (Instrument Control System) has in charge the camera management. The TCS (Telescope Control System) is responsible for the mount and tracking of the telescope. It controls the AMC (Active Mirror Control), the PMC (Pointing Model Camera), the MCS (Mount Control System) and also satefy, interlocks, networking and power infrastructures. The CACS (Calibration Auxiliaries Control System) controls the Auxiliary used for the calibrations of the devices. The ICT component monitors and controls the ICT and Power hardware devices.

INTEGRATION DURING THE CODING

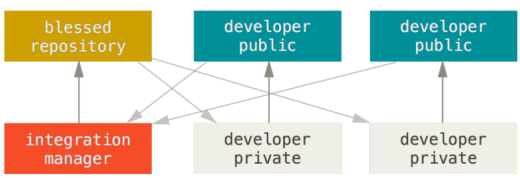

Figure 2: Integration-Manager workflow.

We provide to the developers, who produce code through high level programming, a virtual machine with the environment similar to the real machine on site. We require the developers to test effectively their code before to commit on repository in order to minimize potential errors during integration with other software components. We implemented the integration-manager workflow [13] on Git repo (see Fig. 2) in order to control the software versions and to support the developer who executes the local test of his component which interfaces other components. During the coding period the developer commits in the local (private) development branch. Once the software is ready for the release, the developer pushes the code to the remote (public) development branch. The integration manager clones the public development branches and performs the integration tests. In case of failure the developers have to fix any bug, in order to eventually close with success the integrations tests. Finally the integration manager merges the development branch on the master branch and pushes the codes into the remote master branch (blessed repository). The developers pull the whole last software version from the blessed repository in their development branch to continue the coding for the next release. The interface between the OPC-UA server installed on the hardware device and the OPC-UA client that is part of the ACS component is defined through an ICD (Interface Control Document) created with a predefined format. The ASTRI software has to provide the management of many hardware devices. In order to reduce the coding time, we implemented code generator software (python script) that takes as input the ICD (either in xls or xml format) and provides the ACS component which includes the OPC-UA client. We also implemented the OPC-UA server simulator that get information about the device to simulate from a configuration file provided by the code generator. In this way the developer executes very quickly the local software test without any real hardware. We require also the developer to implement the test for the business logic layer. The test consists of one or more programs that can be repeated by whatever user.

The test suite has to be run by the developer before to commit to the repository the software with the new code. Furthermore ACS includes TAT (Tool for Automated Tests) that is a framework which can run a test suite with only one command and reports the result in a simple and clear way:

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA004

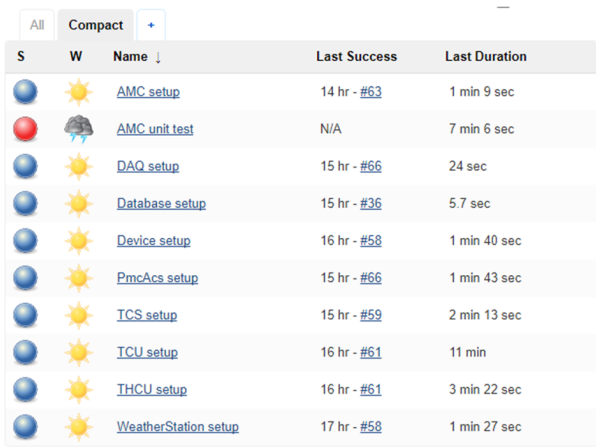

Figure 3: ASTRI continuous integration with Jenkins.

the word PASSED or FAILED is printed to the standard output. Although the ASTRI developers do not push new software with very high frequency, we keep continuously monitored, through a Jenkins server, the software status in order to detect very quickly any breaking build in our development work. We implemented two Jenkins jobs for each software component: the prime compiles the last version on the development branch, the latter runs the unit test suite through TAT. Figure 3 shows the Jenkins page for the ASTRI project. The first column is the status of the build (the build broken is marked with a red bullet); in the second column there is the weather report showing aggregated status of recent builds; follow the name of the job. The integration manager and the developers can display in any moment the details about the continuous integration through a jenkins server installed at the Astronomycal Observatory of Catania.

THE INTEGRATION TESTS

We plan an integration test before any software release. Before we require that the component version candidates for the release locally passed the tests. We execute the preliminary tests on the ASTRI test bed [14]. The test bed is a set of virtual machines which reproduce the same real machines on site. The virtual machines in the test bed have the same configuration (networking, users, operating system, DNS) as the one of real machines so to perform effective tests. All the machines in test bed as well as those on-site have the ASTRI git access read-only in order to update the software version. The test bed has been installed in the INAF IASF Bologna server room, the same server room which will host the CTA Headquarters servers. The goal of these tests is to verify the software setup, and the interactions among software components.

Figure 4 depicts the ASTRI deployment model of the Telescope System as example of the ASTRI software deployment model. The nodes with stereotype *«SL 6.7»* are server machines equipped with Scientific Linux at the version 6.7. During the boot of this machine, we launch the ACS container services in order to exploit the facilities of ACS to run the software in distributed mode. We deploy on the machine slntcs the ACS java component PMC, AMC and THCU (Telescope Health Control Unit) and TCU (Telescope Control Unit) which implement the MCS. The machine slndaq is the camera server and we install there the DAQ software. Also the ACS components DAQ Controller and Camera Controller (written in java), and DAQ Monitor (written in C++) are deployed on this machine. The nodes with stereotype *«device»* denote the hardware where is installed both the firmware and the OPC-UA server to interface the ACS components. The camera device is made of the firmware for the BEE and the FEE (Front End Electronics). We created an installer project for the management of the software components which exploit the ACS services. In particular it provides the functionalities to download the software either from development or master branch. In this project we configure also the parameters to connect the hardware devices, the parameters for the monitoring and the details for the deployment. We install this project and the ACS software components on a dedicated machine where runs the ACS manager that is the responsible for the software deployment. All the other machines are configured to link properly the node where runs the manager. Once the test succeeds on the test bed, the next step is to perform the integration test on-site. During this test we verify the setup procedures on the real servers, the interfaces among the software components, and here we have also to verify the connections between the software and the firmware installed in the hardware devices. This integration test is part of the ASTRI project AIV (Assembly Integration and Verification) plan.

THE SOFTWARE RELEASE

We officially release the software version when all tests are successfully completed on site. The integration manager updates the installer in order to download the specific tested version of software component. Then the integration manager tags and pushes on the master branch all the updated software on the repository. We apply a specific pattern to the tag of the release that is *V.X.Y.Z*, where *X* denotes a major release not always compatible with the previous version, *Y* means a minor release which includes new capabilities and it is compatible with the previous version, and *Z* is used to point the bug fixes. These activities are reported in the release document which provides an overview of that version, the details and capabilities for every component which constitute the version. In this document we also refer to the test reports and user manuals. In addition we published a web based application (see Fig. 5) built on redmine software, to collect the issues, to require new feature or to report any bug. Redmine provides also the functionalities to track the issue status.

CONCLUSION

The procedures presented in this paper, concerning the integration test and software release for the ASTRI project have been applied with success to the first three software

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA004

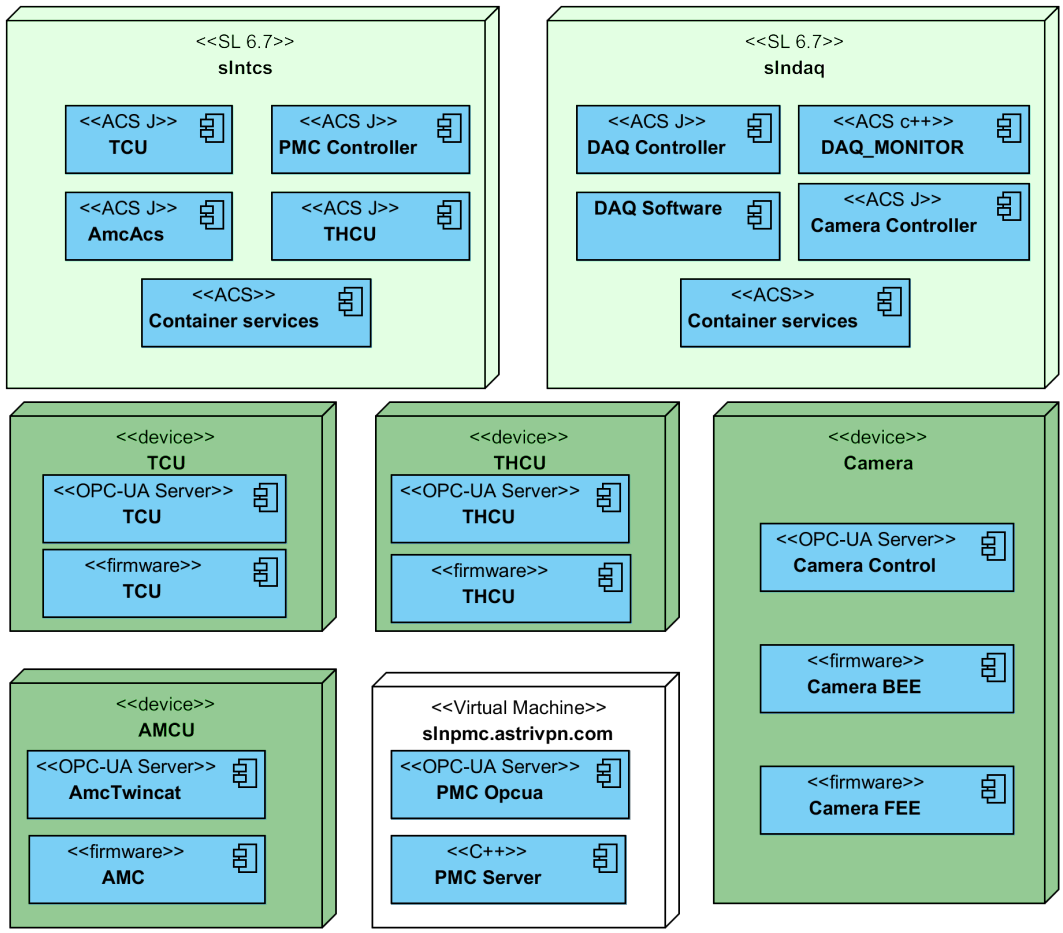

Figure 4: UML deployment diagram of the ASTRI telescope system.

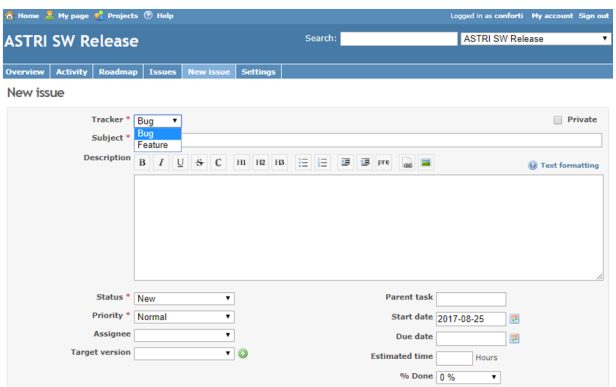

Figure 5: ASTRI web application for the release management.

releases: V.0.1.0 (March 2017), V.0.1.1 (June 2017), V.0.2.0 (July 2017). In those periods we perceived also the benefit coming from this methodology: we provide effective test tools to perform preliminary local tests before the integration, and we test the whole software on a simulated environment. In this way we reduce the testing time on site, and then the dead time for the other AIV actitivies and during the observations when the ASTRI telescopes will be in operation. The second benefit is that in case of failure (either software or hardware) we are able to replace very quickly the current software version with a previous and stable version. Nevertheless we are working to further improve the integration procedure adding such as the methods and tools to perform the code assessment. The ASTRI integration procedure is being used on the ASTRI SST-2M prototype and will be used on the ASTRI mini-array. We may share the ASTRI integration lesson learned with the CTA software team to support the construction phases and the operations.

ACKNOWLEDGEMENT

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

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA004

African Gamma-Ray Astronomy Programme. We gratefully acknowledge financial support from the agencies and organizations listed here: http://www.cta-observatory.org/consortium_acknowledgments. This work was conducted in the context of the CTA ASTRI Project.

REFERENCES

[1] M.C. Maccarone, "ASTRI for the Cherenkov Telescope Array", in *Proc. 35th International Cosmic Ray Conference*, Bexco, Busan, Korea, July 2017.

[2] B.S. Acharaya *et al.*, "Introducing the CTA concept", Elsevier, Astroparticle Physics, *TeV Gamma-Ray Astronomy, Air showers, Cherenkov Telescopes*, vol. 43, pp. 3–18, 2013.

[3] R. Canestrari *et al.*, "The ASTRI SST-2M prototype for the Cherenkov Telescope Array: manufactoring of the structure and the mirrors", in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Montreal, Quebec, Canada, July 2014.

[4] O. Catalano *et al.*, "The camera of the ASTRI SST-2M prototype for the Cherenkov Telescope Array", in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Montreal, Quebec, Canada, July 2014.

[5] M.C. Maccarone *et al.*, "The site of the ASTRI SST-2M Telescope Prototype", in *Proc. 33th International Cosmic Ray Conference*, Rio de Janeiro, Brazil, July 2013.

[6] G. Tosti *et al.*, "The ASTRI/CTA mini-array software system" in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Montreal, Quebec, Canada, July 2014.

[7] V. Conforti *et al.*, "Software Integration for the ASTRI SST-2M prototype proposed for the Cherenkov Telescope Array", in *Proc. 36th Astronomical Data Analysis Software and Systems*, Trieste, Italy, October 2016.

[8] A. Antolini *et al.*, "Telescope Control System of the ASTRI SST2M prototype for the Cherenkov Telescope Array", presented at the 16th International Conference on Accelerator and Large Experimental Physics Control Systems, Barcelona, Spain, October 2017, paper THMPL04.

[9] S. Lombardi *et al.*, "ASTRI SST-2M prototype and mini-array simulation chain, data reduction software, and archive in the framework of the Cherenkov Telescope Array", in *Proc. 35th International Cosmic Ray Conference*, Bexco, Busan, Korea, July 2017.

[10] G. Chiozzi *et al.*, "The ALMA common software: a developer-friendly CORBA-based framework", in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Kona, United States, July 2004.

[11] *OPCUA 2016, OPC Foundation Unified Architecture*, https://opcfoundation.org/about/opc-technologies/opc-ua/

[12] M.C. Maccarone, T. Mineo, M. Capalbi, V. Conforti, and M. Coffaro, "Pre-selecting muon events in the camera server of the ASTRI telescopes foe the Cherenkov Telescope Array", in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Edinburgh, United Kingdom, August 2016.

[13] *GIT distribute workflows*, https://git-scm.com/book/it/v2/Distributed-Git-Distributed-Workflows

[14] F. Gianotti *et al.*, "The ICT monitoring system of the ASTRI SST-2M prototype proposed for the Cherenkov Telescope Array", in *Proc. SPIE Astronomical Telescopes + Instrumentation*, Edinburgh, United Kingdom, August 2016.