ASTRI Mini-Array
Software Integration and Test Model

Table 1.

Prepared by:	Name:	V. Conforti	Signature:		Date:	14/01/2021
Verified by:	Name:	F. Lucarelli	Signature:		Date:	14/01/2021
Approved by:	Name:	G. Tosti	Signature:		Date:	14/01/2021
Released by:	Name:	S. Scuderi	Signature:		Date:	14/01/2021

Main Authors: V. Conforti

Contributor Authors: A. Bulgarelli, F. Lucarelli, G. Tosti, J. Schwarz

Table of Contents

INDEX OF FIGURES & TABLES

Document History		
Version	***Date***	***Modification***
1.0	Nov 5, 2020	First version after internal check of the ASTRI-MA software team

1. Introduction

The **ASTRI Mini-Array (MA)** is an INAF project aimed to observe astronomical sources emitting at very high-energy in the TeV spectral band and perform Stellar Intensity Interferometer of bright stars. The ASTRI MA consists of an array of nine innovative Imaging Atmospheric Cherenkov telescopes that are an evolution of the two-mirror ASTRI-Horn telescope successfully tested since 2014 at the Serra La Nave Astronomical Station of the INAF System of Catania. Each telescope will be equipped with the new version of the Cherenkov Camera based on Silicon photomultiplier detectors. The main science goals of the ASTRI MA in the gamma-ray high energy band, encompass both galactic and extragalactic science. The nine telescopes will be installed at the Teide Astronomical Observatory, operated by the Instituto de Astrofisica de Canarias (IAC), on Mount Teide (~2400 m a.s.l.) in Tenerife (Canary Islands, Spain). The ASTRI MA will be operated by INAF on the basis of a host agreement with IAC.

One of the main components of the ASTRI MA is the software system that provides the tools needed to support the full operation system cycle, from user preparation of an observing proposal to the observation execution; from the analysis of the acquired data to the retrieval of all the data products from the archive. The ASTRI MA software will be developed by INAF and external contractors and will be deployed both at the Array Observing Site (AOS) data centre at Teide (MA online control and data acquisition software), and at the ASTRI data centre in Italy (MA data processing and archiving, user support and data dissemination software).

1.1. Purpose

This document describes the ASTRI MA software system integration, validation and test approach. It is intended as a collection of the procedure that shall be followed to realize the final integrated software release to be used to support the management of the full operation cycle of the ASTRI MA operations. This document will support the software development teams that involves mainly two subjects:

- the INAF team (which includes also other public research institutions such as the Università of Perugia and INFN);
- external contractors.

1.2. Scope

The scope of this document is to provide the ASTRI MA software development teams with the details of the integration and test model for the ASTRI Mini-Array software. The following sections depict the selected integration model and the Integration Manager Workflow that shall be developed considering the repository tree organization.

1.3. Contents

1.4. Definitions and Conventions

1.5 Abbreviations

ASTRI Astrofisica con Specchi a Tecnologia Replicante Italiana
CI Continuous integration

IAC	Instituto de Astrofísica de Canarias
INAF	Istituto Nazionale di Astrofisica
MA	Mini-Array
SDLC	Software Development Life-Cycle

2. Applicable and reference documents

2.1. Applicable documents

[AD1] ASTRI MA Software Development Plan
[AD2] ASTRI MA Glossary and Abbreviations, ASTRI-INAF-LIS-2100-001
[AD3] Software Verification Plan (SVerP) template
[AD4] Software Validation Plan (SValP) template
[AD4] ASTRI MA Verification and validation plan
[AD5] ASTRI MA Quality Assurance Plan
[AD6] ASTRI Mini-Array Project Management Plan

2.2. Reference documents

[RD1] ASTRI MA Top Level Software Architecture, ASTRI-INAF-DES-2100-001

[RD2] URL at the documentation is here: https://docs.gitlab.com

[RD3] ASTRI MA Top Level Use Cases, ASTRI-INAF-SPE-2100-001.

[RD4] https://www.ict.inaf.it/gitlab/astri

3. The concept of the software System Integrator Workflow

This section describes the System Integrator Workflow exploiting the GitLab repository system [RD2]. In particular, this section details the work to be done to test and release the software in a distributed environment as a contributor and an integrator. That is, learning how to successfully contribute code to the ASTRI software and make sub-system maintenance for developers as easiest as possible, and also how to successfully maintain the ASTRI software with a certain number of contributing developers.

3.1 The roles

The software system integrator workflow foresees two main actors:

- The *developer*, who is in charge of the software development and unit testing according to the detailed design document and test plan document;
- The *testing leader*, who is in charge of the software integration test according to the test plan document.

The ASTRI MA organization detailed in [AD1] and [AD6] foresees the following roles in order to perform the test and release activities:

- **Software Coordinator**: responsible for the ASTRI software production and maintenance activities.
- **Release manager**: responsible for managing the complete ASTRI software release delivery life cycle.
- **Maintainer**: the sub work package coordinator;
- **Testing leader**: appointed by the sub work package coordinator (e.g. SCADA) to lead all the test activities (ASTRI team or contractor).
- **Tester**: developer within the sub work package who provides the CI pipeline or performs the manual tests according to the [AD3] [AD4] (ASTRI team or contractor).
- **Developer**: the sub work package who provides source code and tests (ASTRI team or contractor)

The GitLab configuration provides the following roles:

- **Guest**: has limited access to the view of repositories, issues and wiki pages.
- **Reporter**: in addition to the guest grants, it has access to view commits and CI.
- **Developer**: has access to GitLab internal infrastructure & issues.
- **Maintainer**: accepts merge requests on several GitLab projects.
- **Owner**: it defines who owns a specific repository (it is used in ASTRI only to manage the groups configuration at higher level).

See the official page for more information here: https://www.ict.inaf.it/gitlab/help/user/permissions

Because the ASTRI MA software development life-cycle (SDLC) [AD1] prescribes the usage of an iterative incremental development model, the plan is to produce many software releases which incrementally add a small number of specifications to any release. The model envisages multiple iterations of the SDLC until the software release meets all the requirements and the tests have

succeeded. Nevertheless, each iteration shall ensure to produce and release a product which will be tested according to the verification and validation plans [AD3][AD4], and the software quality assurance plan [AD5].

The software architecture decomposes the system into subsystems. For each subsystem, the ASTRI MA sub-work package teams shall provide detailed design, implementation and testing, as well as validation and verification documents. Software integration is necessary to bring together each sub-system into one system, ensuring to deliver the required functionality.

3.2 The workflow

The figure below depicts the Integration Manager Workflow that shall be adopted for the ASTRI MA software integration.

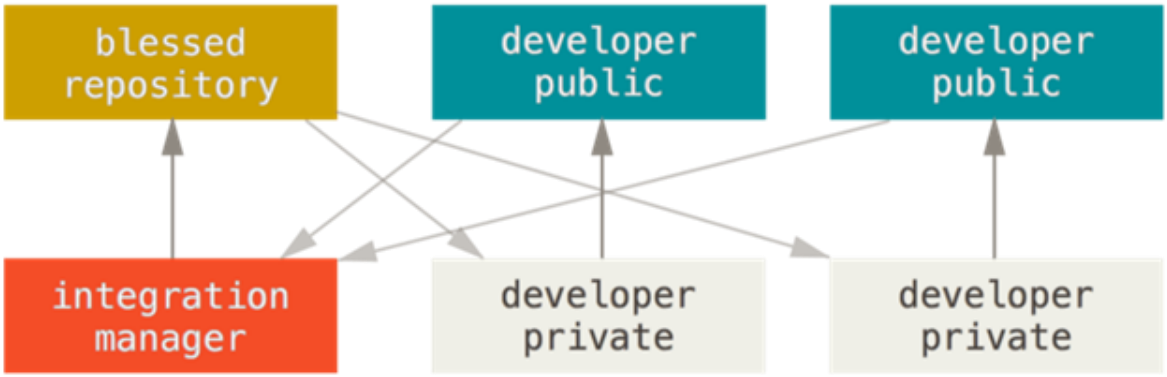


Figure 1.

This is a very common workflow with hub-based tools. One of the main advantages of this approach is that developers can continue to work applying their own SDLC, and the maintainer of the main repository can pull-in the changes at specific time. Contributor developers do not have to wait for the project to incorporate their changes. The developers can work at their own pace and methodology on their own development branch.

The Integration Manager workflow envisages the following repository areas:

- *developer private*: it is the space to support the development activities;
- *developer public*: it is a dedicated space where the developer moves a specific software version once it is ready for the integration tests;
- *blessed repository*: it is a reserved space to move the software version once it is ready for the official release.

3.3 The locations / git branches

The Integration Manager workflow requires an infrastructure (either hardware or software), properly configured, to support the integration tests and the releases. The git repository allows the developers to implement for their own software the version control in a very effective way. The development teams are free to implement whatever workflow for the software development. The scope of this document is

to provide a common view to the sub-system teams to perform the testing and release activities. Therefore, we intend to exploit the git branches to apply the workflow. Specifically, the *master*, *test* and *dev* branches will be used for this objective.

4. Integration-Manager workflow applied to the ASTRI Mini-Array

The ASTRI software is under version control through the git-based GitLab software [RD4]. Here ASTRI is a gitlab group which contains 5 sub-groups:

- SIMULATIONS
- SCIENCE-USER-SUPPORT
- DATA-PROCESSING
- ARCHIVE
- SCADA

Each group contains all the repositories needed to perform the version control of the specific sub-system software. All the repositories have been configured with the developers, who provide the source codes, and the maintainer who has in charge the management of the repository, such as Merge Request (MR) and tagging.

The developer uses the assigned repository to submit the code under version control. The developer pushes the files in a no-protected branch, that means any branch but not the *dev*, *test* and *master* (that are protected branches). The developers can use any other additional branch which also triggers their own separate testing pipeline on every merge, without any approval of the maintainer.

When the software is ready to be evaluated for the sub-system internal release, the developer sends a MR on the *dev* branch to the maintainer. Once the maintainer approves the request, then the *dev* branch has a new version and the CI pipeline runs the unit tests and the static code analysis on this new version. For each repository, the teams can create as many releases in the *dev* branch as they want.

The integration test phase requires, according to the test plan, all the repositories be ready with the following documentation:

- unit and functional tests execution status for each sub-system, which corresponds to the list planned in the Requirement Verification documents;
- static code analysis report in SonarQube and code coverage;

Therefore, when the software of a repository is ready, the developer sends a MR on the *test* branch to the maintainer of the repository. The maintainer at least checks that the functional tests status in the CI pipeline for the mentioned change-set corresponds to the generated report (i.e., faults filed for all the failed tests, and quality criteria defined for current release are met). Beside that, he checks the code quality report for the change-set, which is going to be merged, meets the quality criteria adopted for the current release (making sure that the report generated is for this particular change-set). Once the maintainer approves the request then he performs the merge and tags properly the version (see section 6).

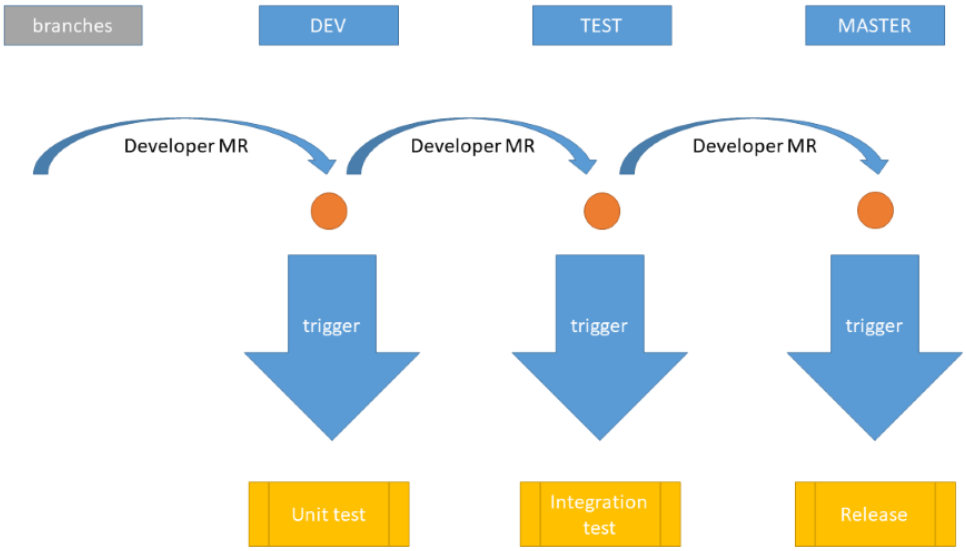
When all the repositories are pushed in the *test* branches, then the testing leader starts the integration tests. The tests in this phase can be either automatic or manual according to the requirement verification documents. The tests shall be performed on the test bed and eventually on the production environment (on-site or offsite).

If any test fails then the testing leader rejects the version and asks the teams responsible for the code that caused the bug to patch the code and provide as soon as possible a new version in the *test* branch[1]. The testing leader evaluates the bug-fix and decides to execute completely or partially the integration tests.

If the tests pass and all the requested documents are provided (such as test and quality reports) then the software is ready to be released. In this case the testing leader sends a MR on the *master* branch to the maintainer of the repository. Once the maintainer approves the request then the software has a new official version. Concerning the merging activities, the maintainer is responsible to solve any conflict in order to keep ahead the last tested version.

During the integration test activities, the *test* and *master* branches are frozen. The developer can continue the development exploiting any no-protected branch.

Most of the sub-systems use one repository, but more than one repository is accepted as long as a sub-system repository provides complete and consistent information to assemble the software version from the other derived repositories. The next sections provide more detail concerning the activities timeline based on specific repositories. The figure below shows all the activities triggered on push to the specific branch as described above.



[1] This new version must contain only a commit to fix the bug, and not any other features or recent implementations which could introduce new integration issues with other components.

Figure 2.

4. Organization of ASTRI and sub-systems

The ASTRI software is composed by a set of subsystems which work together to implement the ASTRI software requirements and use cases [RD3]. The ASTRI system, described in the architecture design document [RD1], will be deployed both on site and offsite as a whole product, and therefore we need a repository which is the main access point to the software. The astri-mini-array repository in Gitlab is devoted to be the main ASTRI software access point. It includes all the consistent information that is required for an ASTRI software version, linked to the other repositories, in order to fully define the software release. Nevertheless, the ASTRI team is composed of sub-task teams in charge of producing sub-systems. Any team can follow their own organization and lifecycle with the support of the AIV (Assembly Integration and Verification) team as long as they respect the overall ASTRI workflow.

Concerning the development, the subsystems teams can use different development methodologies with the following commonalities:

- The interfaces between sub-systems are governed by the interface control documents.
- The common software (such as libraries, frameworks, infrastructures) shall be agreed before the development.
- The sub-system teams cooperate with the SW ENG team for the testing and release activities.
- The sub-system teams observe the release plan and the prescriptions of this document.

5. The timeline

The release plan for the specific release foresees four main milestones detailed in the figure below. The milestone M1 requires the complete documentation concerning the requirement documents, the verification documents and the design documents. This milestone is preliminary to the development time, and then we expect at this time that the interfaces are well defined. Therefore, the version of interface-control-document repository shall be approved and tagged in the master branch.

The milestone M2 requires the software ready for the verification at sub-system level, then the sub-system teams shall provide the software and the tools for the verification (such as Jenkins and SonarQube). The sub-system repositories shall deliver the software as versions in the dev branch, as well as in the astri-mini-array repository.

The milestone M3 is devoted to close the integration tests (either automatic or manual), then we expect at this time the integration test reports. The sub-system repositories shall provide the software version in the test branch as well as the astri-mini-array repository.

The milestone M4 is devoted to the release, then the sub-system repositories shall provide the software version in the master branch as well as the astri-mini-array repository.

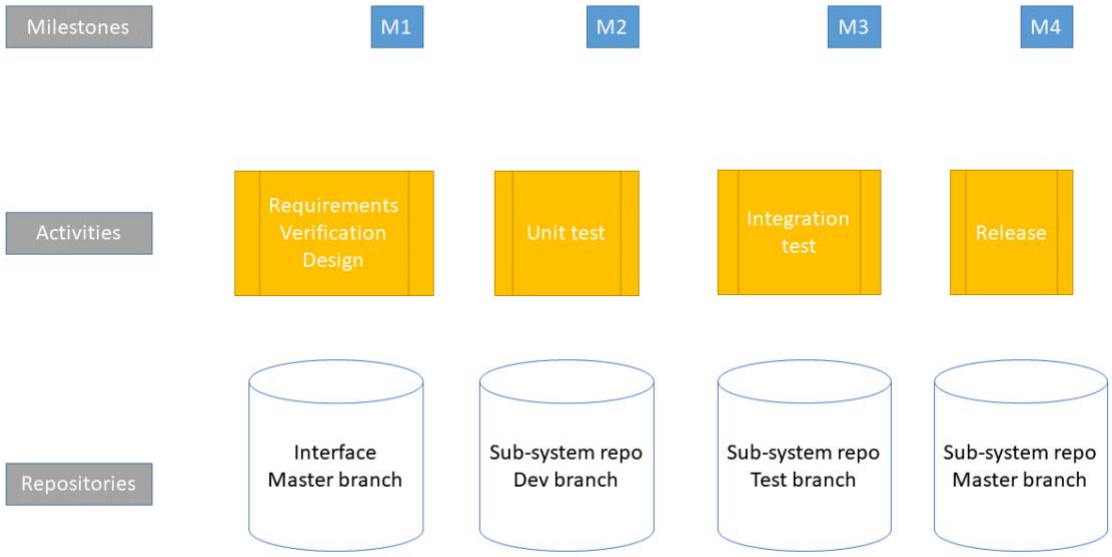


Figure 3.

6. Version tagging

Concerning version tagging, ASTRI Mini-Array follows the software versioning rules detailed in this section.

Version numbers shall follow the semantic versioning standards1, where versions have three integer parts: MAJOR.MINOR.PATCH.

- MAJOR version number, which is incremented when there are incompatible API or functionality changes;
- MINOR version number, which is incremented when functionality is added in a backward compatible manner;
- PATCH number, which is incremented when there is a backward-compatible bug fix applied.

For non-release versions, the version string shall contain more information such as the exact commit which can be identified in the revision control system. For example: mypackage-1.0.dev728, or mypackage-1.0.1-post7+git1fecb48 (the number of commits posterior to the version 1.0.1 tag, plus an optional git hash).

Released versions shall be tagged in the revision control system, with stable releases as branches with at least the MAJOR and MINOR version numbers in the branch name, so that older versions can continue to be maintained via patches without interfering with the development of newer versions.

It is recommended that for each software release, a unique identifier (e.g., a DOI) is generated to make the software release reproducible. This item is to be finalized.

7. The ASTRI Mini-Array software repository

This repository is the main access point to the ASTRI software and shall contain:

- A way to download the specific subsystems version which form the release;
- A way to configure the system (CDB, components-containers, and so on);
- A way to run the software.

In order to download the software, a specific target “download” may be inserted in the Makefile which provides the instructions to download the right version from the right branch (*master* for the release, *test* for the test activities) for any sub-system.

The configuration can be provided in XML format, like the CDB of ACS.

Concerning the software execution, specific scripts can be written, e.g. to start/stop ACS manager and the containers.

At time of writing this document is not well defined as the final strategy. This is intended as a preliminary proposal to discuss.

8. The repository organization

The repo tree is defined according to the ASTRI management plan. The git system selected is GitLab, and it is available at the following URL: https://www.ict.inaf.it/gitlab/astri. The table below details the groups compounding the ASTRI system. Each group shall not contain any sub-group but only the repositories (minimum number to reach the goal). Each repository shall be created with three main branches properly configured: *master*, *test* and *dev*. In addition, tokens shall be created to support the automatic tests.