INAF
ISTITUTO NAZIONALE
DI ASTROFISICA

IN PARTNERSHIP WITH
SKAO

TETIS
TEchnologies for Telescopes and
Instrument control Software

USC VIII

Testing SKA control software for quality assessment and early anomaly detection

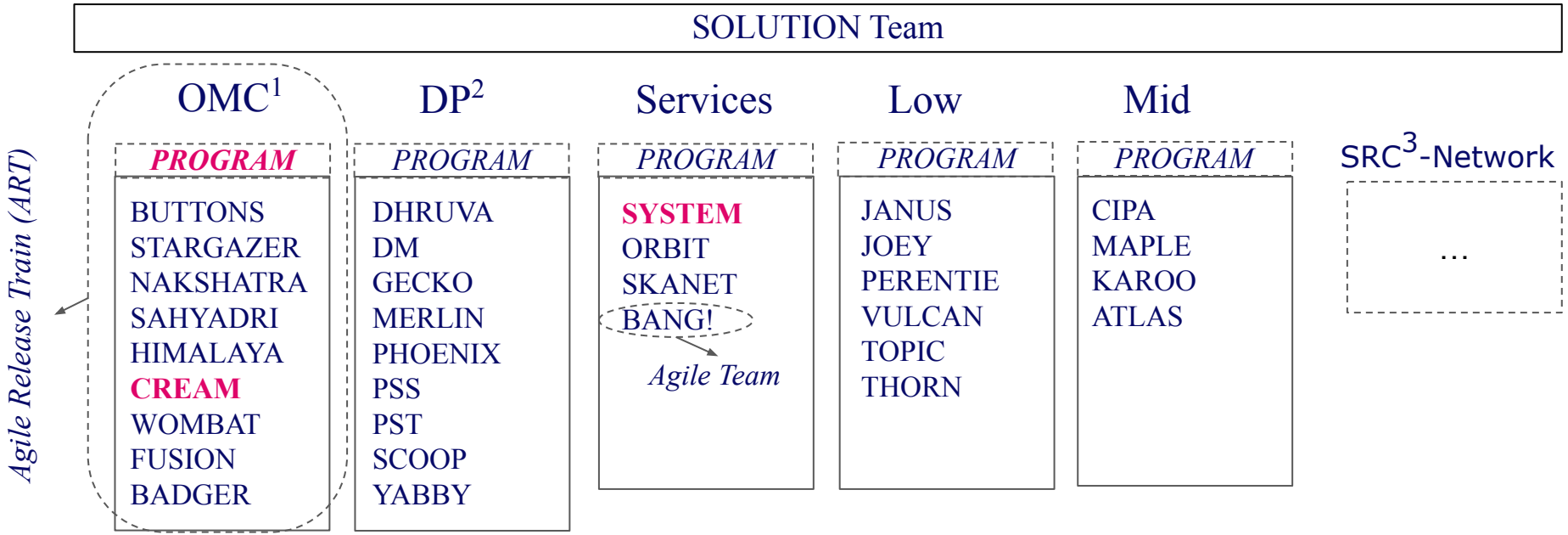
Stefano Di Frischia, Gianluca Marotta on behalf of the INAF SKAO Group

The SKA software engineering group

The SKA software engineering group

- more than 200 people involved from 15 different countries
- organized with Scaled Agile Framework (SAFe)
- work is iteratively planned every 3 months (Program Increment - PI)

SAFe® PROVIDED BY SCALED AGILE



[1] *Observation Management and Control* [2] *Data Processing* [2] *SKA Regional Centers*

The INAF Group

Program Team

CREAM TEAM

SKAO SYSTEM

OATs	Valentina Alberti	UI/UX[1] Expert (OMC Program)
OAA	Carlo Baffa	CREAM Deputy Product Owner
	Elisabetta Giani	CREAM Developer
	Gianluca Marotta	CREAM Developer
OAAb	Stefano Di Frischia	CREAM Developer
	Matteo Canzari	CREAM Developer
	Matteo Di Carlo	SYSTEM Developer
	Mauro Dolci	Scientific Responsible
IRA	Teresa Pulvirenti	Contract Management

2

CSP — Local Monitoring and Control

TARANTA TANGO ON WEB

IC@MS

System Infrastructure

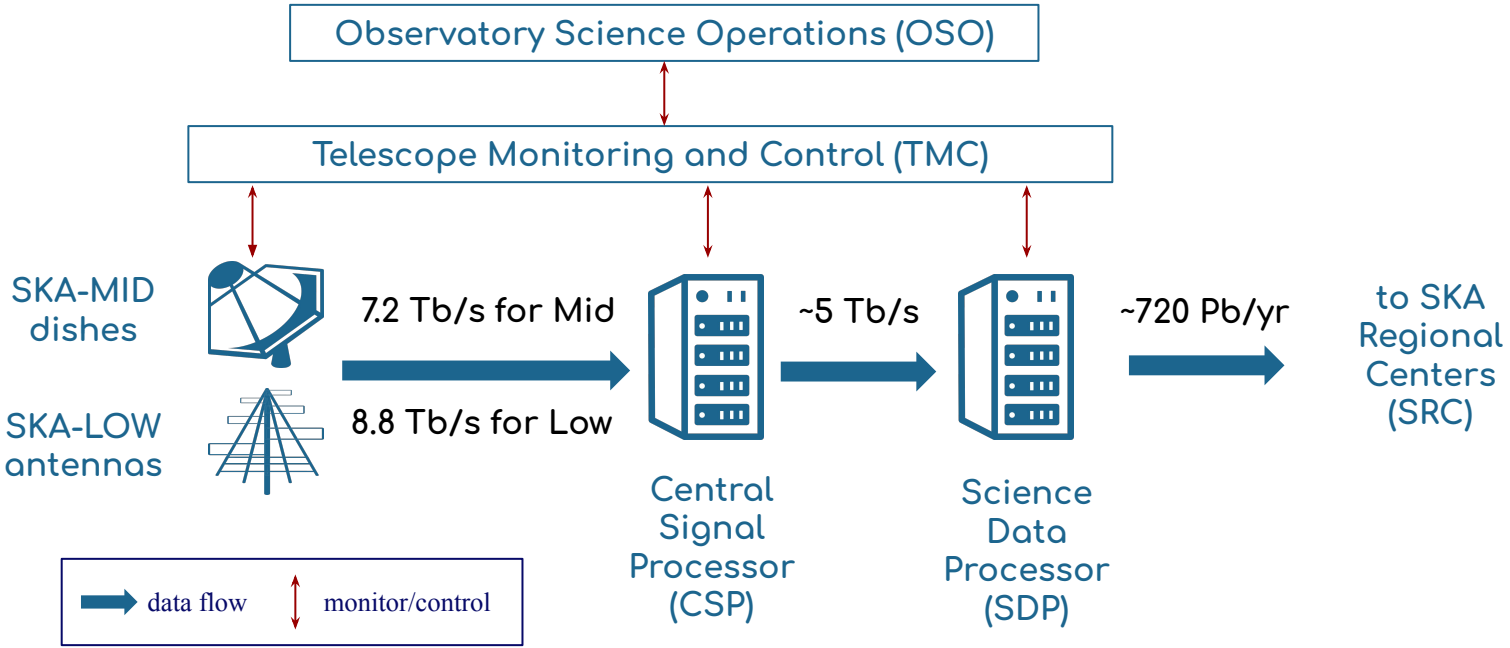
N.B. CREAM and SYSTEM are also composed by people coming from other countries and companies/institutions

[1] *User Interfaces/User eXperience*

[2] *Central Signal Processor*

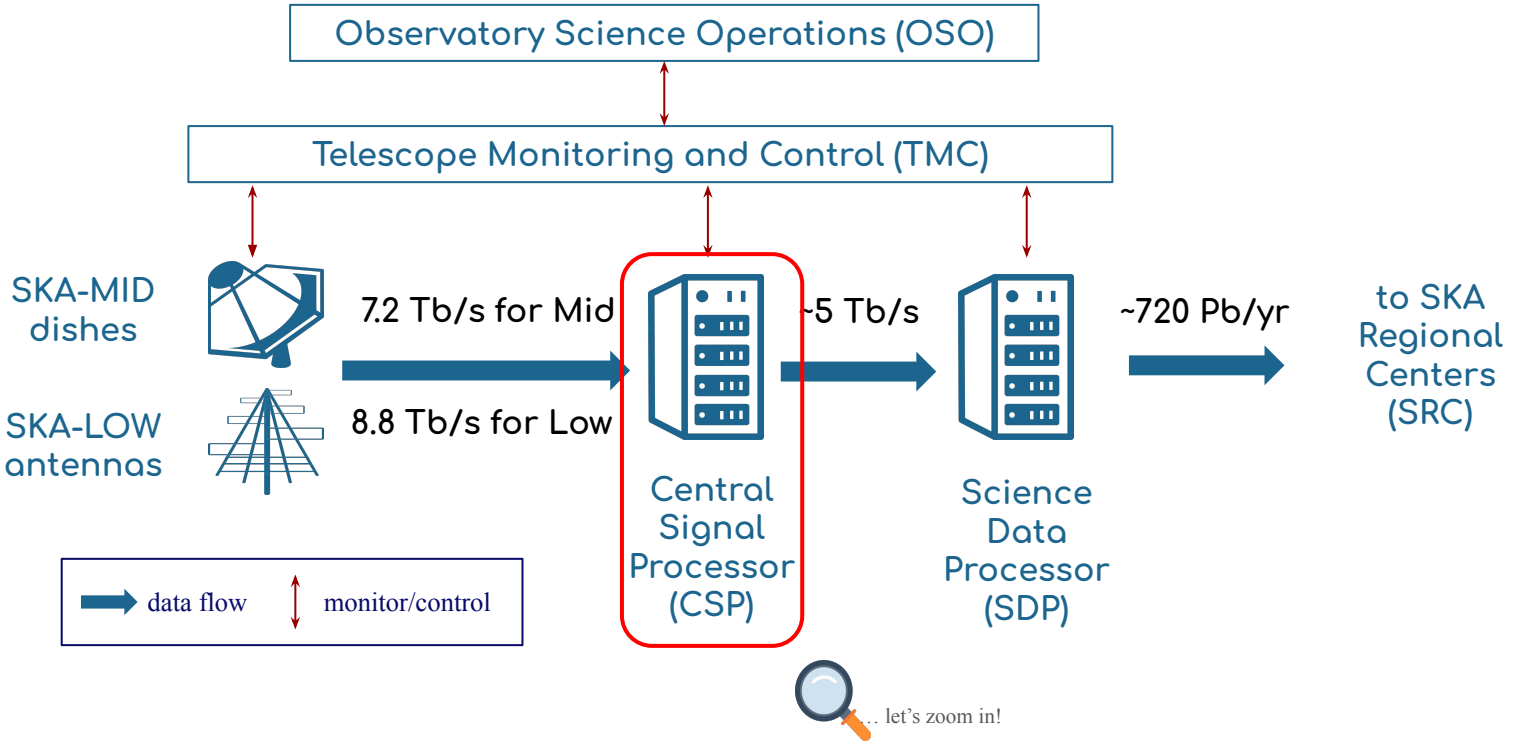
The SKA Data Flow and Control System

… a very simplified view

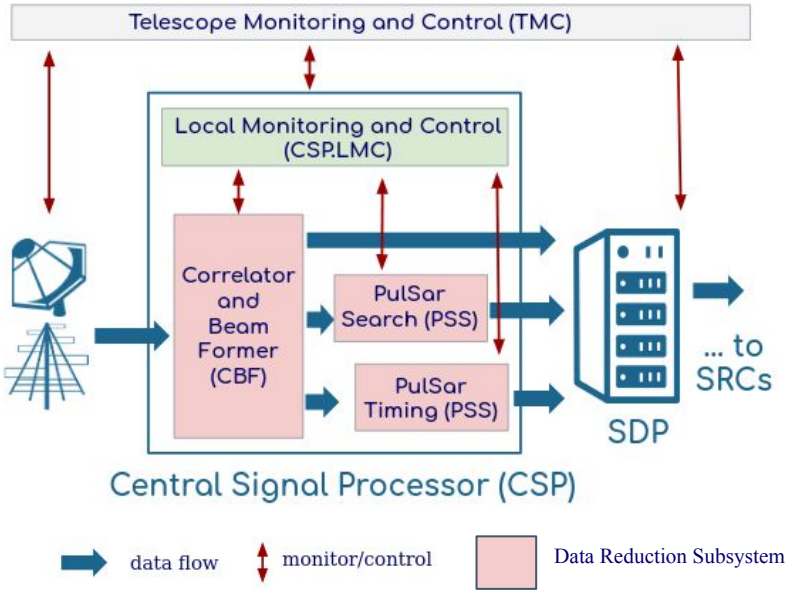


The SKA Data Flow and Control System

… a very simplified view



The Central Signal Processor

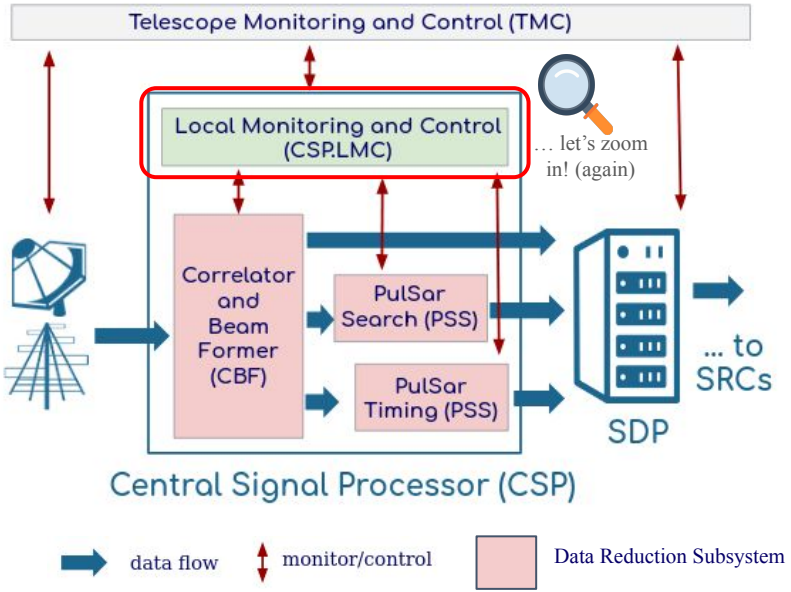


CSP is composed of three main subsystems:

- the **Correlator and Beam Former (CBF),** to creates the visibilities and the data beams;
- the **Pulsar Search (PSS)**, to perform an all-sky pulsar search survey;
- the **Pulsar Timing (PST),** to measures the frequency of the pulsar candidates

CSP.LMC provides the *interface* to TMC *without exposing CSP internal complexity.*

The Central Signal Processor



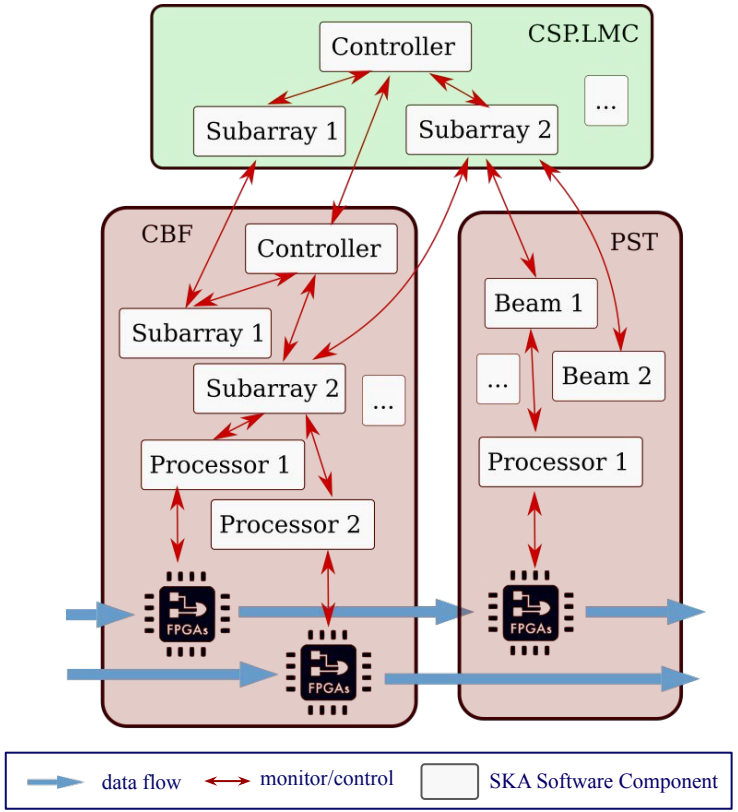
CSP is composed of three main subsystems:

- the **Correlator and Beam Former (CBF),** to creates the visibilities and the data beams;
- the **Pulsar Search (PSS)**, to perform an all-sky pulsar search survey;
- the **Pulsar Timing (PST),** to measures the frequency of the pulsar candidates

CSP.LMC provides the *interface* to TMC *without exposing CSP internal complexity.*

The CSP Local Monitoring and Control (CSP.LMC)

… a very simplified view



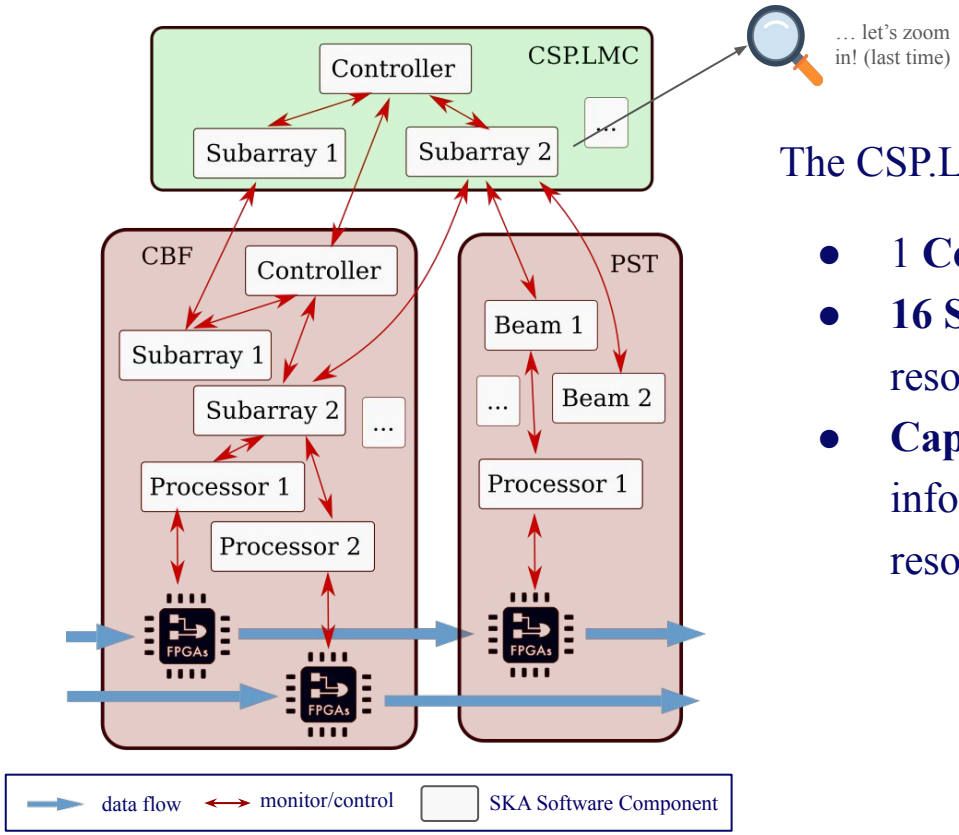
The CSP.LMC is composed by:

- 1 **Controller,** i.e. the primary point of access for CSP
- **16 Subarray**, representing subsets of the telescope resources that can be used for one observation
- **Capability** devices, apt to monitor and report to TMC information and statistical data about specific CSP resources (e.g. CBF processors, PST Beams)

CSP **L**ocal **M**onitoring and **C**ontrol

The CSP Local Monitoring and Control (CSP.LMC)

… a very simplified view



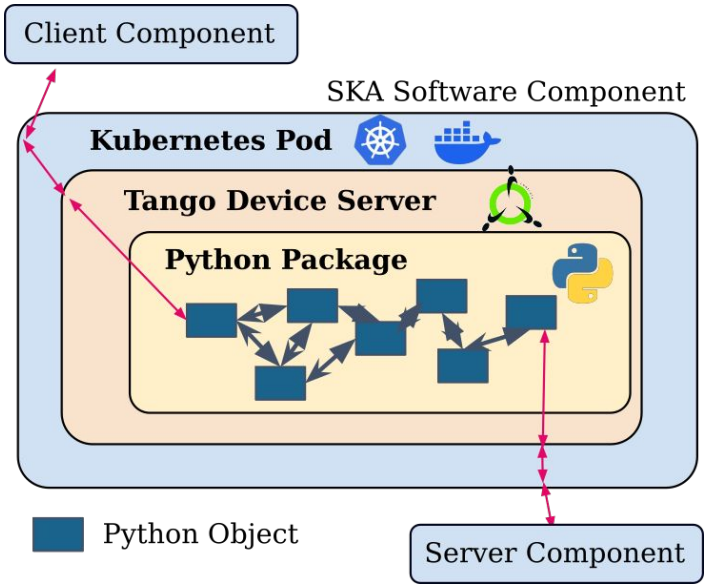
The CSP.LMC is composed by:

- 1 **Controller,** i.e. the primary point of access for CSP
- **16 Subarray**, representing subsets of the CSP resources that can be used for one observation
- **Capability** devices, apt to monitor and report to TMC information and statistical data about specific CSP resources (e.g. CBF processors, PST Beams)

CSP **L**ocal **M**onitoring and **C**ontrol

The SKA Software Component

Software is **executed by** a **Tango Device Server** process, that runs in a **Docker container** that is **orchestrated** in a cluster **by Kubernetes**



TANGO

"a free open source device-oriented controls toolkit for controlling any kind of hardware or software and building SCADA[1] systems"[2].

docker

"is an open platform for developing, shipping, and running applications [...] enables you to separate your applications from your infrastructure[3]"

kubernetes

"is a portable, extensible, open source platform for managing containerized workloads and services …[4]"

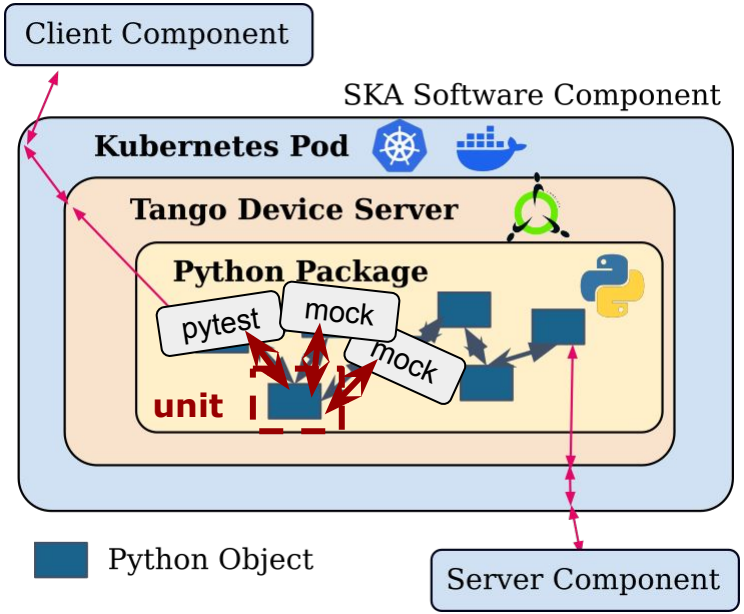
* for simplicity only one server/client is reported

[1] Supervisory Control And Data Acquisition [2] https://www.tango-controls.org/ [3] https://docs.docker.com [4] https://kubernetes.io/docs

How are we testing?

Tests are performed with a *multi-level strategy:*

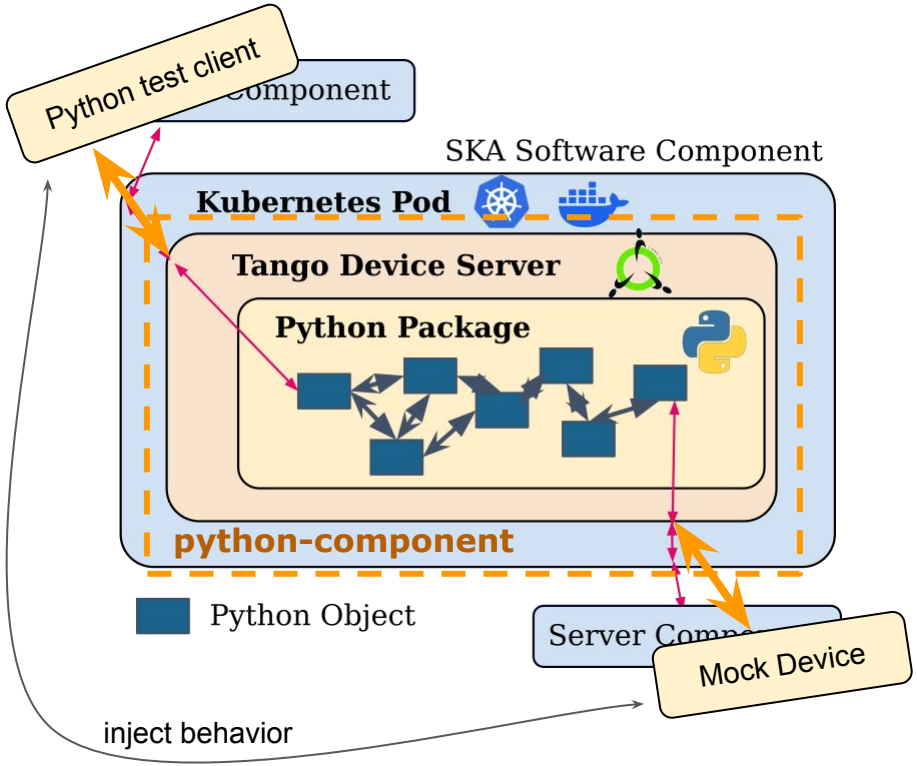
- *unit* tests

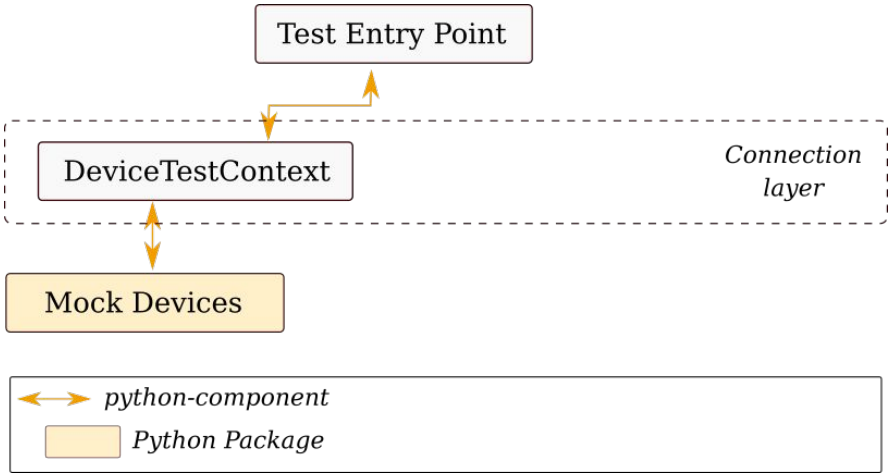


How are we testing?

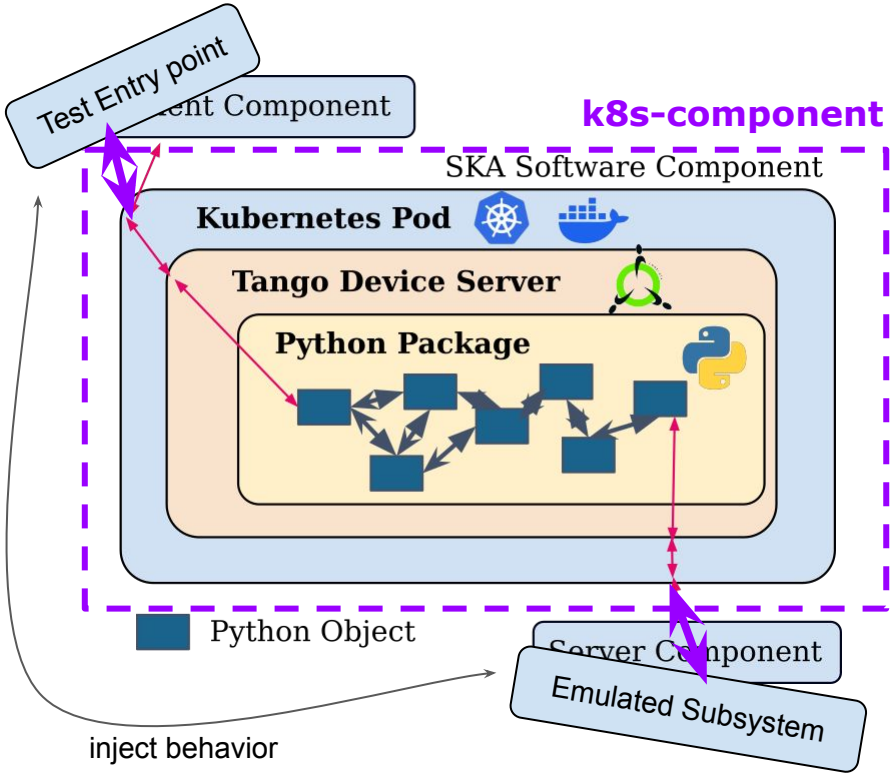
Tests are performed with a *multi-level strategy:*

- *unit* tests
- *python-component* tests



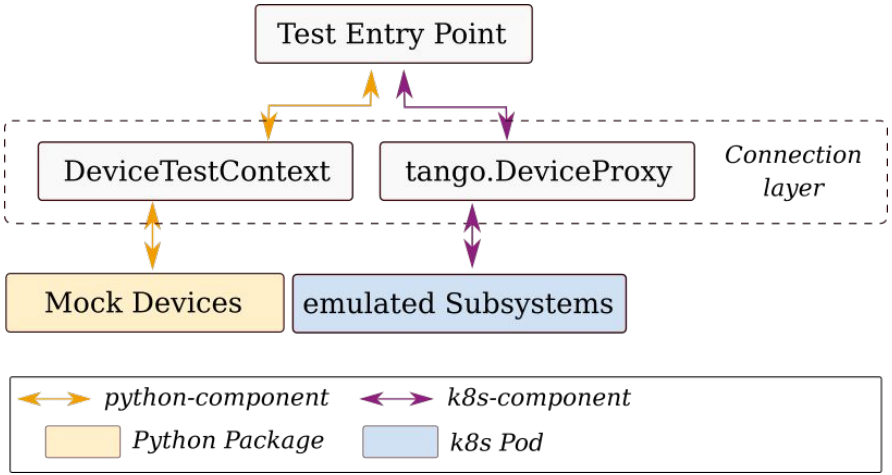


How are we testing?



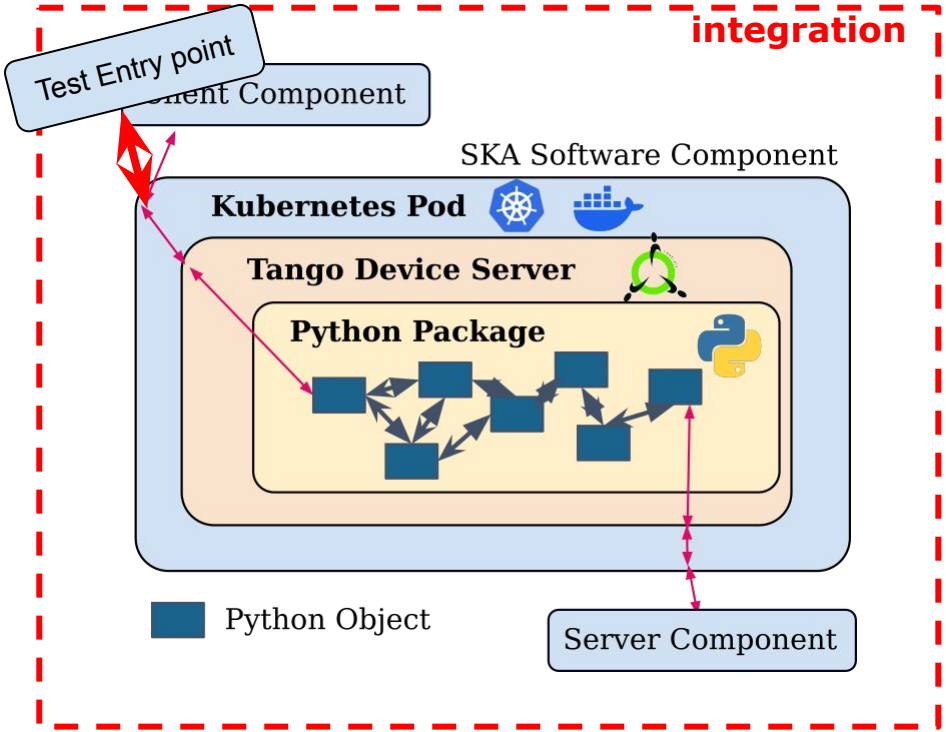
Tests are performed with a *multi-level strategy:*

- *unit* tests
- *python-component* tests
- *k8s-component* tests



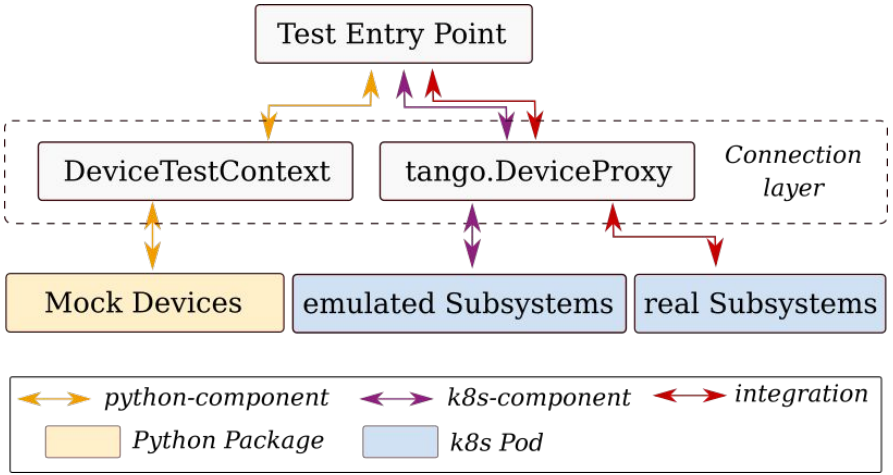
How are we testing?

The CSP.LMC Testing Strategy



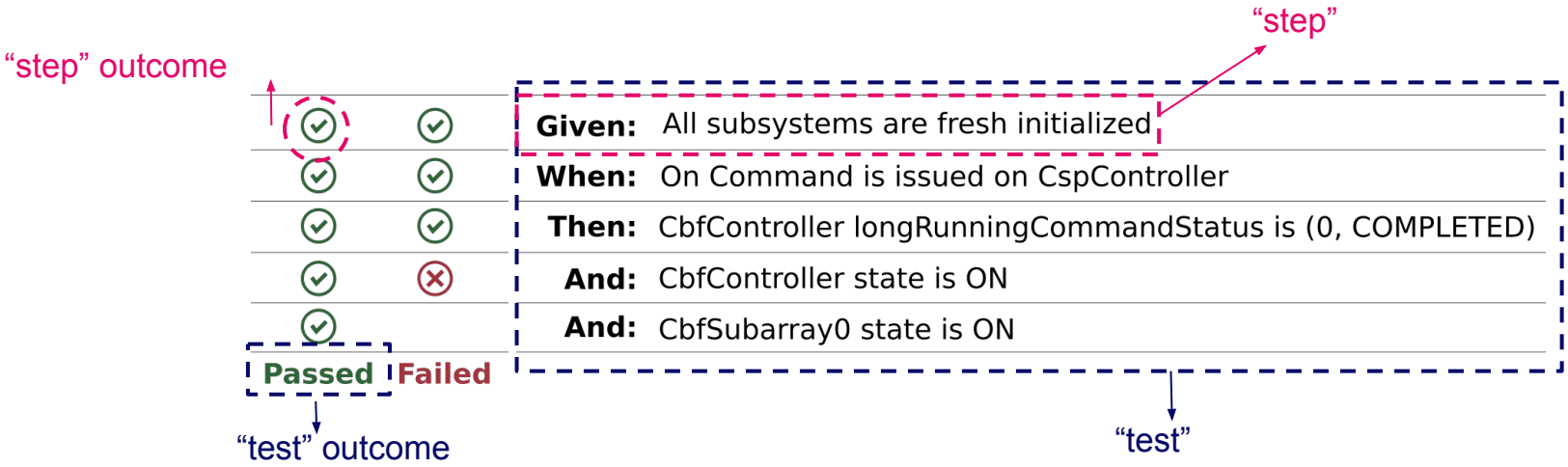
Tests are performed with a *multi-level strategy:*

- *unit* tests
- *python-component* tests
- *k8s-component* tests
- *integration* tests



How are we testing?

Integration and *component* tests follow the *Behaviour Driven Development (BDD)* approach. They are written in the *Gherkin* language.



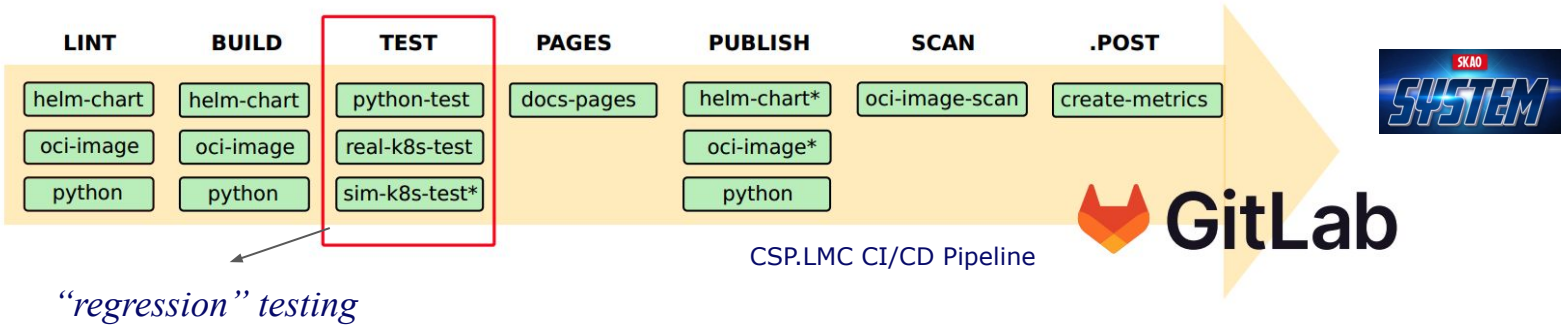
Each "step" is translated to a specific Python function and can be utilised in different tests

Continuous Integration/Continuous Delivery/Deployment

Continuous Integration/Continuous Delivery/Deployment (CI/CD) refers to development practices:

- single source repository for each component;
- automated build;
- automated testing;
- **every commit** should build on an integration machine (with tango/kubernetes)

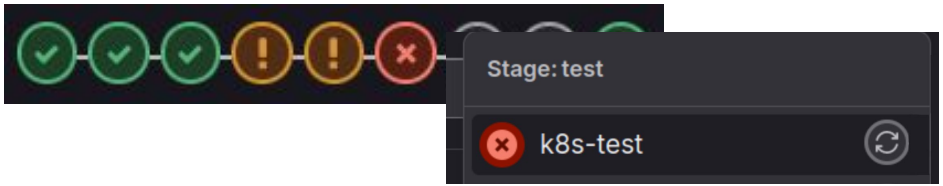
CI/CD practices are ensured by the use of **Gitlab pipelines,** based on System Team templates.



What if a test fails… sometimes?

“**Regression testing** is performed at least every time code is committed on any branch in the source code repository. This should be ensured by the CI/CD pipeline”

SKAO Software Testing Policy and Strategy



Two possible causes for “red pipeline”:

1) the change in the code disrupted an existing functionality
2) one or more tests has failed in a non-reproducible way

*“test **flakiness**”*

How to deal with test flakiness?

It seems that *few* solutions are possible:

1) ~~block the development until the test has been fixed~~
2) ~~log as bugs/failures and put in backlog~~ → "Flaky failures" are difficult to reproduce!
3) ~~remove the flaky tests~~ → This is *ignoring bad evidence!*
4) retry until pipeline is green *(yes, we all do…)*

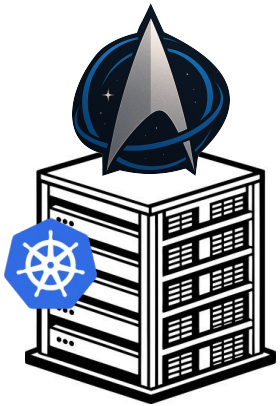
we decided to try a *new solution*…

collect and **analyze** the test results!

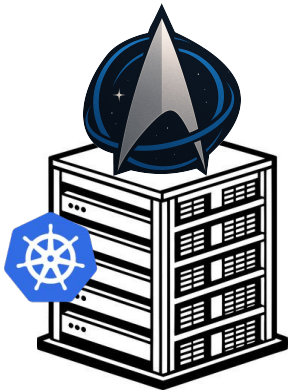
HW Infrastructure for SKA Software @ INAF

Finanziato dall'Unione europea
NextGenerationEU

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA



@INAF-OAA

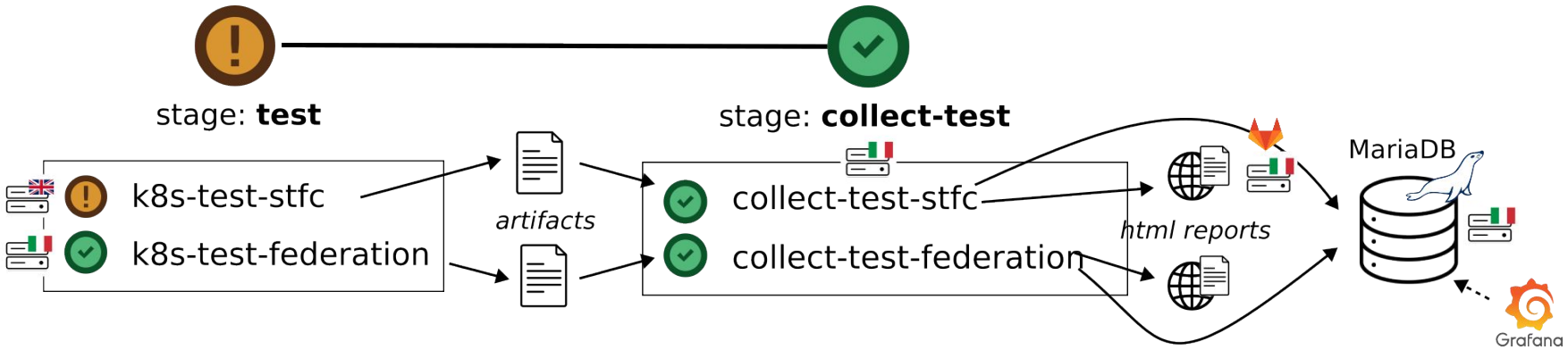


@INAF-OAAb

Federation Clusters

- made available free of charge
- provide kubernetes
- hosted and managed by INAF-OAA and INAF-OAAb members
- hosts Cream Team VMs for development
- hosts CSP.LMC CI/CD pipelines

Collecting tests result using Gitlab CI/CD



- **Periodic pipelines** every 6 hours (4 times a day)
- Only **two stages** in periodic pipelines (no build – **only released versions**)
- k8s–tests run on **two** similar **facilities**
- Jobs in stage "**test**" are **allowed to fail**
- Artifacts coming from test execution are **parsed** and **collected** by stage **"collect-test"***

MariaDB

Database structure 1/2

Pipeline jobs

job_id	pipeline_id	facility	job_time	subsystem_type
1040503	187283746	STFC	2025-09-08 18:03:32	real
…	…	…	…	…

Product versions

job_id	product_name	version
1040503	ska-csp-lmc-low	1.0.1
1040503	ska-low-cbf	1.0.3
1040503	ska-pst	1.0.0
…	…	…

Test executions

id	job_id	test_name	test_start_time	setup_outcome	setup_duration
98	1040503	test_on_command	2025-09-08 18:03:32	passed	0.0012
…	…	…	…	…	…

teardown_outcome	teardown_duration	test_duration	test_outcome	error_traceback
passed	0.865	2,52	passed	…
…	…	…	…	…

Database structure 2/2

Test executions

id	job_id	test_name	test_start_time	setup_outcome	setup_duration	…
98	1040503	test_on_command	2025-09-08 18:03:32	passed	0.0012	…
…	…	…	…	…	…	…

Test steps

id	test_id	step_index	step_keyword	step_name	step_outcome	step_duration
489	98	0	Given	CSP is initialized	passed	0.604
…	…	…	…	…	…	

Test logs

id	test_id	log_index	time	log_level	code_source	thread	message
768	98	0	2025-09-08 18:03:32	INFO	base_wrapper.py:135	MainThread	low-csp/subarray/01 available …
…	…	…	…	…	…		

Addressing test flakiness - the hunt for hidden bugs

The Global Fail Rate (GFR) is *a* **metric** that reports, for each version, the **combined quality** of **control software** and **testware.**

$$GFR = \frac{\#\ tests\ \text{failed}}{\#\ tests\ \text{executed}}$$

The best approach is to do the exploration of the DB as a **collective effort** during **Explore Test DB sessions**

- To be planned every 2/3 weeks
- Iterative process to improve the collecting of data results
- Items in backlog to lower the GFR

we take advantage of **Grafana dashboards** and **HTML Reports**

Grafana Dashboard - the hunt for hidden bugs

db: ska_csp_lmc_mid_test_records | scenario: csplmc subarray and fsp capability reports correct attributes after configuration 1 and 2 fsp no pstbeams

example: 01-"Configure CBF 2ndFSP ver5.json"-'mid csp cbf/fsp/02'-2-("ON",)-("UNKNOWN",)-"CORR", "CORR", "CORR", "CORR", "PST BF", "PST BF", "PST BE", "ON", "DISABLE", "DIS

csp_version: 1.1.0 | step_name: | test_id: 27700

Global Fail Rate over CSP.LMC versions -- click on link for details

CSP.LMC version	total_tests	failed_tests	Global Fail Rate (%)
1.1.0	20605	1329	6.45

Facility Fail Rate over CSP.LMC versions -- click on link for details

CSP.LMC version	facility	total_tests	failed_tests	Global Fail Rate (%)
1.1.0	INAF-Federation	11440	374	3.27
1.1.0	STFC	9165	955	10.4

Fail Rate over test sessions

seq_id	job_id	Global Fail Rate (%)	go_to_report
1	10415614384	6.15	http://10.17.17.13:310...
2	10415891958	3.08	http://10.17.17.13:310...

Most Failed Scenario Outline -- click on link to show failures per examples

scenario_outline	failed_count
csplmc subarray and fsp capability reports c...	166
csplmc subarray successfully release all reso...	90
csplmc subarray and vcc capability reports c...	84

Most Failed Scenario examples (per selected Scenario Outline)

scenario_example	failed_count
01-"Configure CBF 2ndFSP ver5.json"-'mid c...	57
01-"Configure CBF vis 5.json"-'mid csp cbf/f...	55

Most Failed Steps (per selected Scenario)

step_name	failed_count
CspSubarray01 has assigned no PstBeams a...	50
Configure command is executed with SUCCE...	6

Absolute Most Failed Steps

step_name	failed_count
CspSubarray01 has been configured with no ...	243

Anomaly detection

Anomaly detection is the foundational first step in building a robust AI-based **predictive maintenance** program for SKA.

1. Anomaly detection
2. Diagnosis and Prognosis
3. Predictive Maintenance

- Integration tests simulate the intended, correct interactions between different components.
- The data generated during these (successful) tests is invaluable for establishing a "healthy" operational baseline.
- Anomaly detection models are trained to recognize this normal state, so any deviation can be flagged.

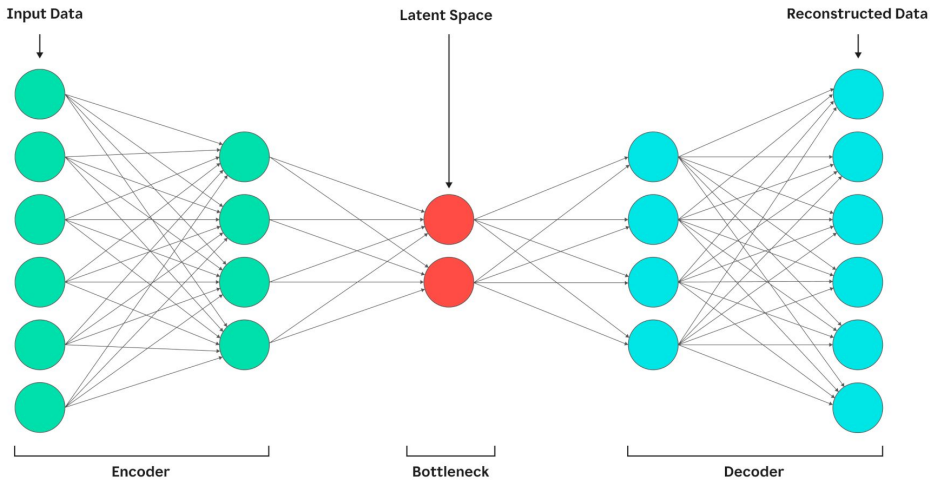
Failed Test ≠ Anomaly !

Autoencoders for anomaly detection

Once periodic test results have been collected, and after an accurate **Data Ingestion** and **Data Preprocessing** phase, a ML model can be trained in order to perform **Unsupervised Anomaly Detection**.

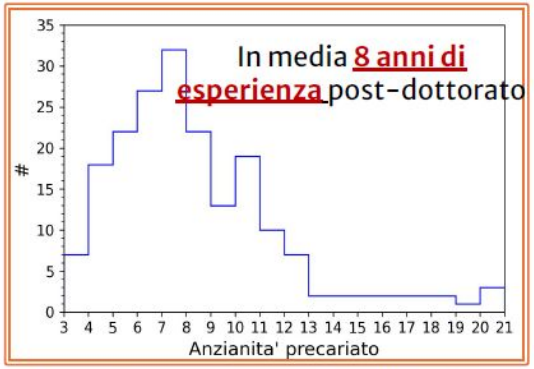
Deep Autoencoders:

- latent space can be considered a representation of the system standard baseline
- perform well on high-dimensional input points
- can capture nonlinear cross-system dependencies

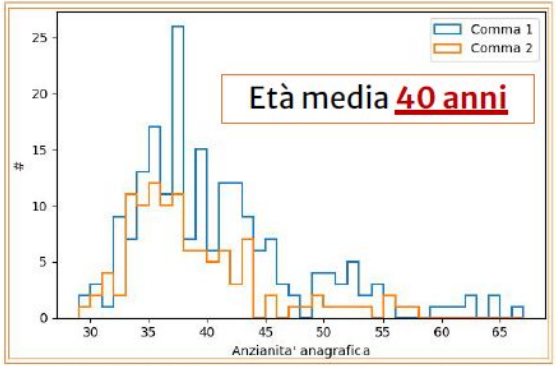


La situazione del personale precario in INAF è INSOSTENIBILE!

1.200 Tempo Indeterminato Vs **650** precari: più di 1 precario ogni 2 persone di ruolo



Plot di un campione rappresentativo dei precari INAF al 31/12/2024



Dei 650, **287** possono essere stabilizzati:
173 tramite chiamata diretta (comma 1)
114 tramite concorsi riservati (comma 2)

Entro l'anno, l'attuale situazione determinerà l'esodo di > 100 lavoratori altamente qualificati e il MUR se ne lava le mani

È **URGENTE** che INAF **PROCEDA ORA** con le **STABILIZZAZIONI TRAMITE MADIA:** unica soluzione per questa emergenza

RETE STABILIZZANDI INAF

Molti colleghi (972) hanno già firmato, per sostenerci e aggiungere il nome alla lista del QR,
contattaci a retestabilizzandi1.inaf@gmail.com

Thank you for your attention!

Testing SKA control software for quality assessment and early anomaly detection

Stefano Di Frischia, Gianluca Marotta on behalf of the INAF SKAO Group