VERA C. RUBIN OBSERVATORY

ENGINEERING METHODS FOR VERIFICATION & VALIDATION

Holger Drass, Austin Roberts for the V&V team

NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

Overview

1. Verification System Architecture
2. Testing at the Summit
3. Backup Material

Motivation

Why would this presentation be for you?

Are you collecting or analysing data from the
TMA, M1M3, M2, hexapods or rotator or any other system?

You are part of verification and validation team!

Advantage:

Mutual benefit – Supporting teams gain systems experience – Project get artifacts

Why, Why, Why….

Why do we do all of this? Why now? - We are "nearly" ready to observe…

We, as a project, want to make sure that we built
what we were saying we are going to build.
For that we do by verification.

And well…

We want to make sure what we built is doing what we said it will do.
To prove that we do validation.

You might ask: What did we say and agreed on back then? (I was not even there…)
The sayings became system requirements. Therefore we do V&V on the requirements.

Why, Why, Why….

We need to keep a memory on how we fulfilled the 10000 things we were going to do.
No exaggeration here: 9602 Reqs + 1090 Hazard mitigations.
Some reports structured by topic help a lot.

Our V&V is not only for us.
People are trusting us and give us lot of money to support us.
They (our agencies) like to see reports on how it went with their spendings.

Input of the reports comes from the Test Case executions →

Test Cases are our friends

Verification System Architecture

Verification Definitions

Verification Ticket - A specific item that must be verified in order to verify a requirement or part of a requirement.

Coverage - The Verification Ticket is traced to a Test Case in our Jira Verification System

Verified by VE - The Verification Ticket is verified once the linked Verification Ticket is verified. Typically done for requirements flowdown.

Descoped - Either the requirement was descoped or the verification of the requirement has been descoped.

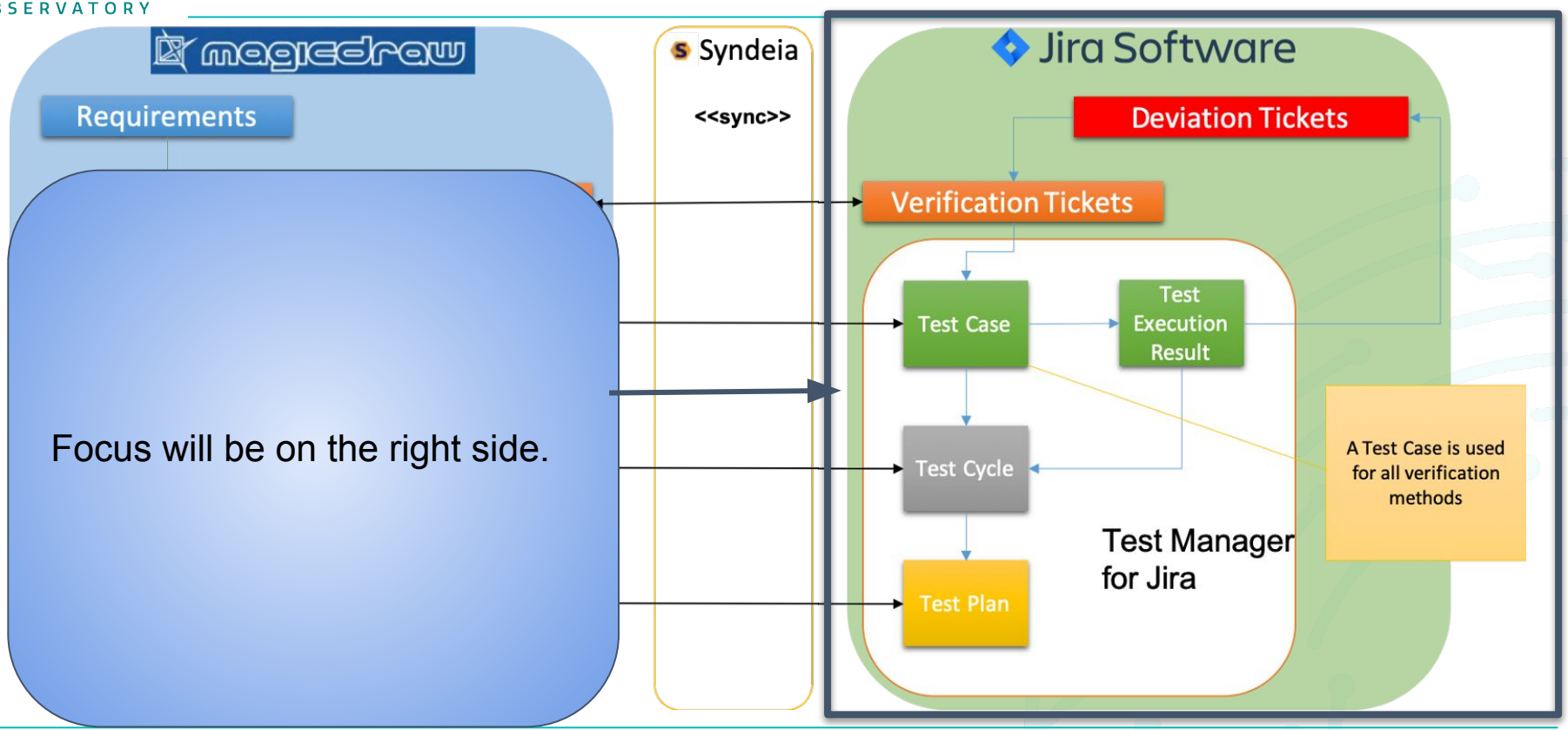
“Not Covered” Status - The Verification Ticket is not traced to all the required Test Cases.

“Covered” Status - The Verification Ticket is traced to all required Test Cases.

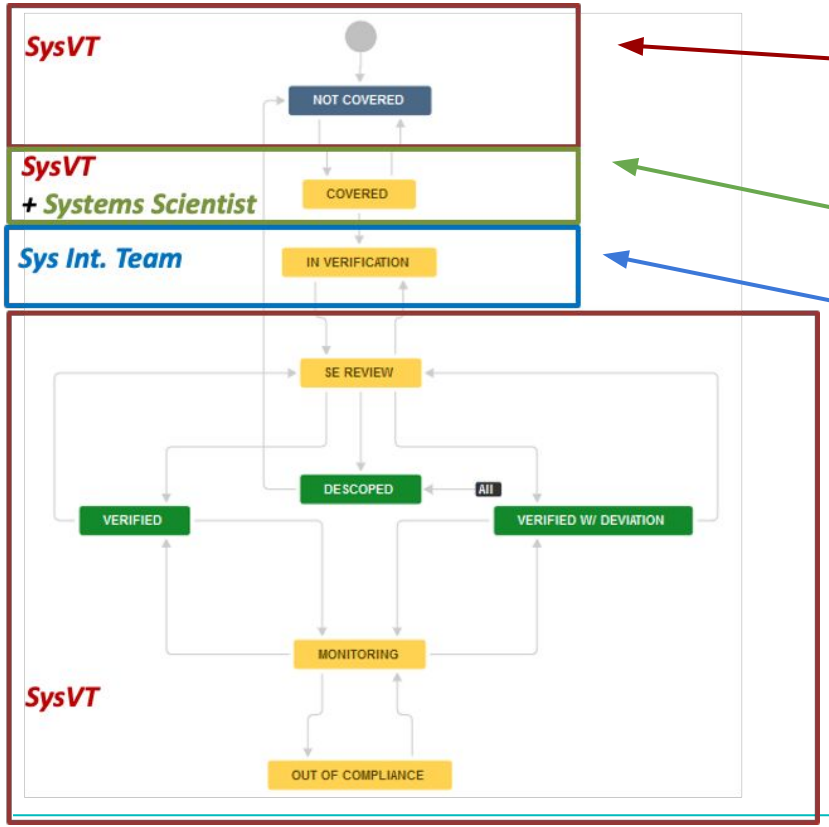
Test Case - A Test Case in Jira either represents a verification artifact from a vendor or a procedure carried out by Rubin Observatory.

FRACAS - Failure Reporting, Analysis and Corrective Action System (MIL-HDBK-2155)

Verification Architecture



Verification Ticket Workflow – What does it looks and who is doing what?

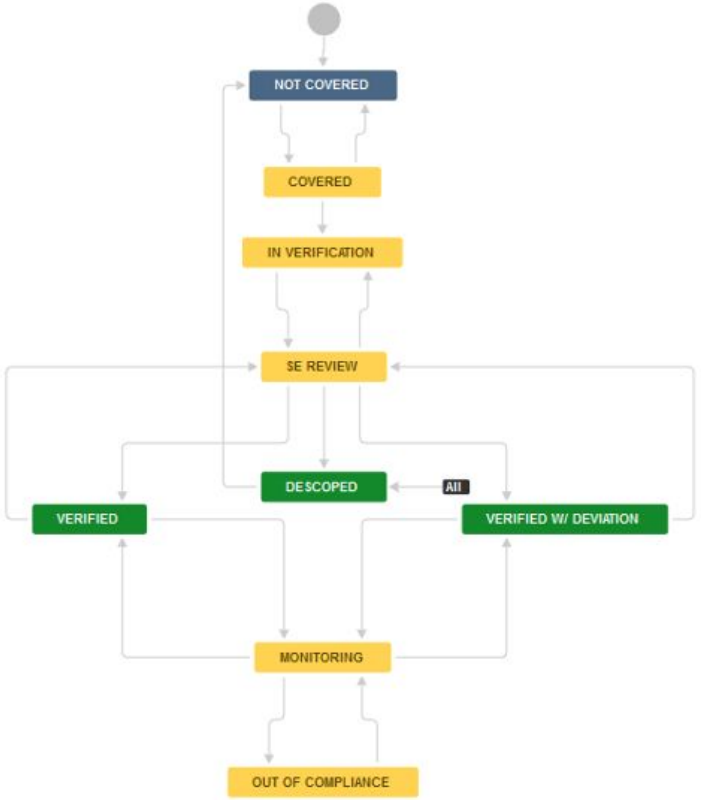


- System Verification Team (SysVT) responsible for upfront planning
  - Definition of Verification Events (Test Plans)
  - Mapping to Verification Phases (Test Cycles)
  - Definition of Verification Procedures (Test Cases)
- SysVT works with the Systems Scientist and Engineers on detailed procedures (Monday's V&V meeting)
- System Integration (Sys Int.) Team performs the activity (Verification Event) and analyzes the results
- SysVT
  - Reviews the results and makes the determination on the sufficiency of results
  - Tracks verification progress
  - Monitors verification compliance after acceptance

What is Verification Ticket Workflow good for?

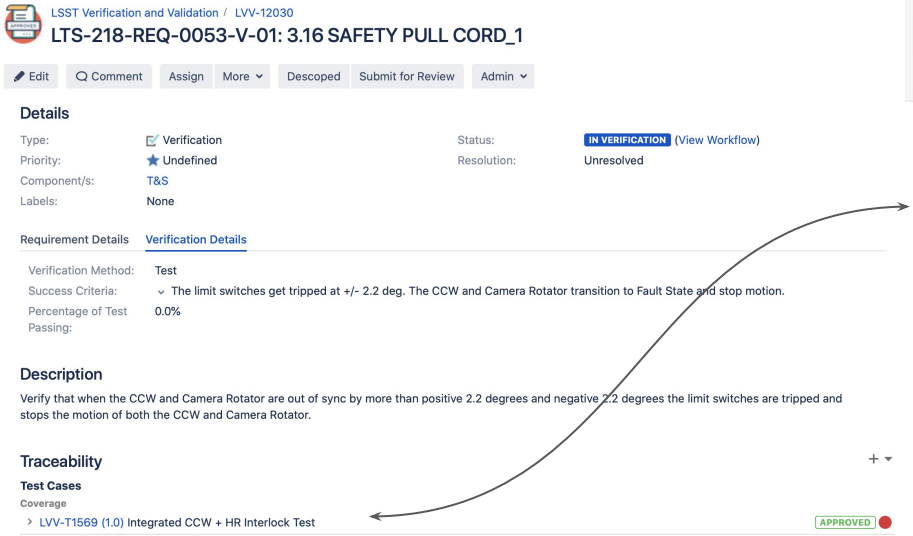
- Jira workflow hold us true the Rubin Observatory verification process
- Evaluates links from Verification tickets to Test Cases and the verification results of the Test Cases
- Evaluates Verification tickets verified by the verification of other Verification tickets and their approval status.
    - **Note:** Component (lower) level requirement can verify Subsystem or System (higher) level requirement. This is very much preferred.

      But (as it comes to reality) the other way is possible, too.
- Evaluates Deviation Request tickets generated during verification and their approval status



Note: Verification Elements Coverage

The Verification tickets in Jira are either linked to another Verification ticket with the Verified By relationship or to a Test Case with the Coverage relationship. Once all the relationships have been created an agreed on the Verification ticket is then transitioned to the Covered status. These relationships can be seen bi-directionally.



Verification Plan Development

In the Jira **Test Plan** we define the overall scope of the verification event by populating the following fields.

- Objective
- System Overview
- Verification Environment
- Entry Criteria
- Exit Criteria
- PMCS Activity
- Observing Required?
- Document ID
- Verification Artifacts
- Overall Assessment (populated for the test report after execution)
- Recommended Improvements (populated for the test report after execution)

Verification Cycle Development

In the Jira **Test Cycle** we define any **configuration differences** between the other Test Cycles within the Test Plan by populating the following fields.

- Description
- Software Version / Baseline
- Configuration
- Planned Start Date
- Planned End Date

Test Case Development

The **Test Cases** details and Test Script are defined to capture what is required in order to execute the Test Case and the detailed step-by-step procedure and expected results. The Test Case starts in Draft status, then to Defined once it is ready for review, and then Approved after review.

Test Case details

- Objective
- Precondition
- Predecessors
- Jupyter Notebook Link
- Critical Event (Involving glass)
- Verification Type
- Verification Configuration
- Unit Under Test
- Required Software
- Test Equipment
- Safety Hazards
- Test Personnel
- Required PPE
- Associated Risks
- Postcondition
- Vendor
- Reuse for Maintenance

Test Script details

- Step
- Test Data
- Expected Results
- Example Code
- Conditional Step?

Summary: Verification process

Plan

1. Develop **Verification Plan** (Test Plan in Jira) to capture scope of a Verification event
2. Develop **Verification Cycle**(s) (Test Cycle in Jira) to capture phases of event and differing configurations between phases
3. Review Verification Elements to determine what requirements to verify
4. Populate Verification Elements details
5. Trace Verification Elements to Verification Procedures (Test Case in Jira)
6. Organize and sequence Test Cases into **Test Cycles**

Implement

7. Develop details and steps of Test Cases
8. Create Jupyter notebook to perform the Test Case
9. Produce Test Plan / Test Report document

Execute

10. Schedule event in JIRA summit calendar and procure equipment, personnel, etc.
11. Execute Test Cases and capture execution results in Jira
12. Follow up on the findings by creating any Deviation Requests, FRACAS reportable failures, and Bug tickets
13. Produce updated Test Plan / Test Report with results, assessment, and recommendations

Verification Plan Generation

Test Plan / Report PDF generated from information from the Test Plan, Test Cycle(s), Test Case(s), and Verification Elements.

How a Test Plan / Report PDF is generated

- Python scripts to query Jira REST API
- Latex file produced
- Document appears on lsst.io
- PDF generated for DocuShare archival
- SIT-COM uses a SIT-COM Test Report (SCTR) handle

The Need For SCTR Documents

- Commonality across project
  - DM developed the report generation architecture and uses the same process
  - EPO was accepted by funding agencies using the same process
- The project must provide a Technical Data Package (TDP) to the funding agencies and Rubin Operations
  - The TDP must be in a static format and be able to be easily delivered to and reviewed by the stakeholders
  - The Jira based verification system is dynamic so we must be able to baseline the content
- The verification results and details must be in the test case executions
  - This can be
    - directly in Actual Result field
    - attached to a test step
    - attached to overall execution
    - A link in the execution to a controlled repository such as DocuShare

Testing at the Summit

Test Cases – Our Friends

It takes a moment to get familiar with then but than they are pretty easy to handle and have nice feature traceability features.

Let's go a small tour.

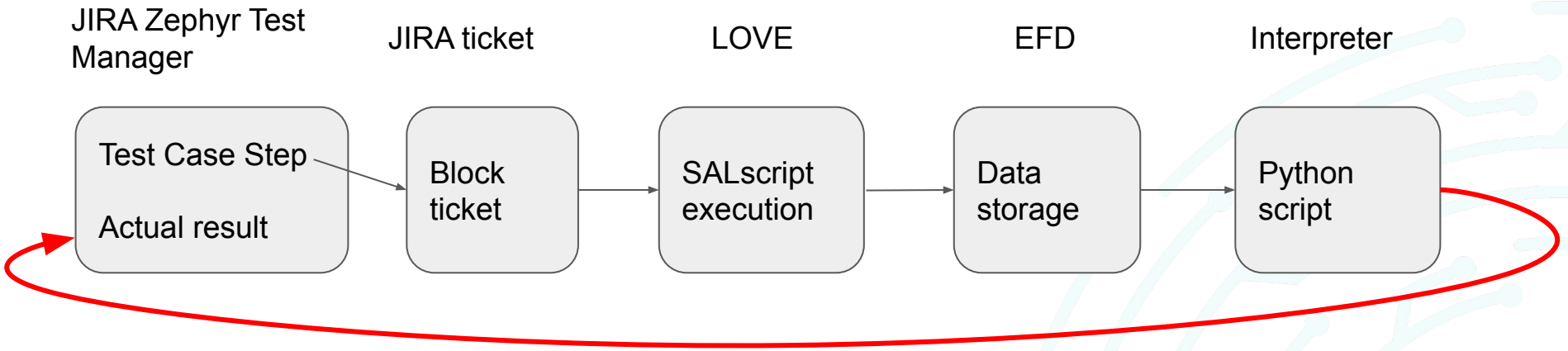
Actually we have two ways to fill a test case execution

in our JIRA test manager Zephyr:

Test Case vs Test Case Execution.
A test case is a template. Each time you execute a Test Case, you start an execution or go on filling an existing one. You can have as many executions as you want.

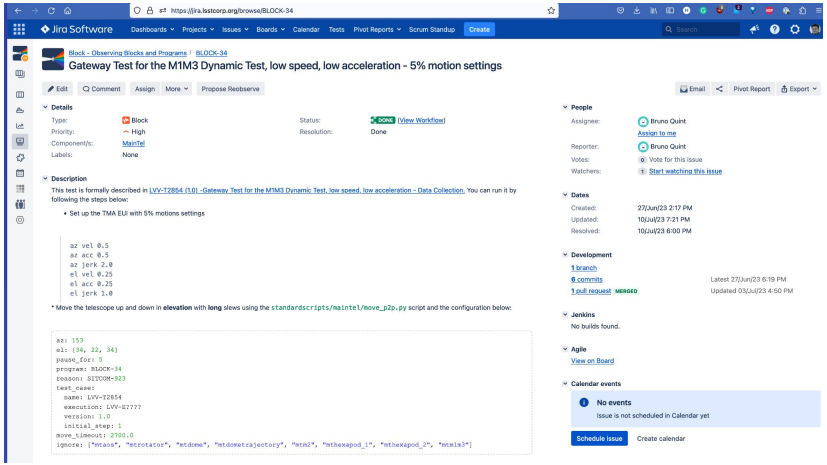
Zephyr – Jupiter Notebook/ Love SALScript – EFD – Zephyr	Zephyr – Block ticket – Love SALScript – EFD – Interpreter – Zephyr
	

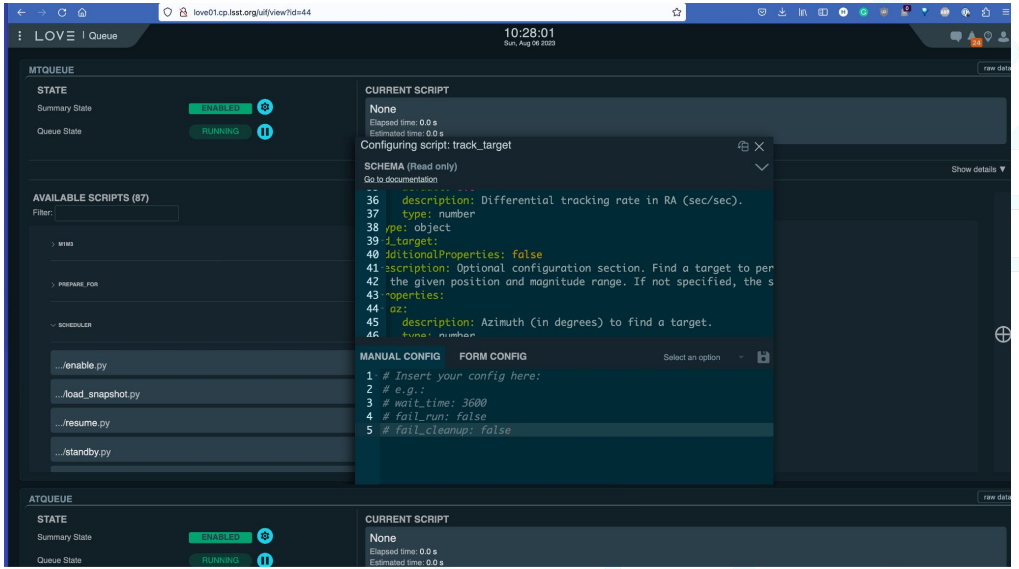
NEW V&V Testing Workflow



Block Usage

The block holds a copy of the test step(s) and some instructions for the SALScript configuration





Q&A

Backup material

Jupyter Notebook Development

Jupyter Notebooks are created to perform the command and control through SAL following the Test Script.

Synchronous Interlock CCW/Rotator Integration Test

This notebook performs a synchronous motion interlock scenario integration test between the Camera Cable Wrap (CCW) and the Rotator with the Camera Cable Wrap tracking the Rotator. It includes enough boilerplate to allow the test to run at any time by getting current time information from the pointing and computing appropriate coordinates to slew.

```
[ ]: import logging
     import yaml

     import os
     import sqlite3
     import pandas as pd

     import numpy as np
     from matplotlib import pyplot as plt
     import astropy.units as u
     from astropy.time import Time
     from astropy.coordinates import AltAz, ICRS, EarthLocation, Angle, FK5
     import asyncio
     from lsst.ts import salobj
     from lsst.ts.idl.enums import ATPtg

     from astropy.utils import iers
     iers.conf.auto_download = False
```

```
[ ]: test_message = "Interlock Rotator_CCW Integration Test"
```

```
[ ]: d = salobj.Domain()
```

```
[ ]: script = salobj.Controller("Script", index=1)
     rot = salobj.Remote(d, "Rotator")
     mtptg = salobj.Remote(d, "MTPtg")
```

```
[ ]: await asyncio.gather(rot.start_task,
                          mtptg.start_task,
                          script.start_task)
```

```
[ ]: await asyncio.sleep(1.)
     await salobj.set_summary_state(mtptg, salobj.State.ENABLED)
     await salobj.set_summary_state(rot, salobj.State.ENABLED)
```

Set Low Velocity and Move Through Positive Interlock

```
#The velocitySet command gets accepted but the moveConstantVelocity command gets error.
#moveConstantVelocity has not been implemented

rot.cmd_velocitySet.set(
    velocity=0.02,
    moveDuration=115.0
)

await asyncio.sleep(0.1)

await rot.cmd_moveConstantVelocity.start(timeout=30.)
```

```
[ ]: await asyncio.sleep(1.)

     print("Move to 2.3 deg position")

     await rot.cmd_positionSet.set_start(angle=2.3,timeout=10.)

     await rot.cmd_move.start(timeout=30.)

     print("Reset the Rotator and CCW")
```

Move to Zero

```
[ ]: print("Move to 0.0 deg starting position")

     alt = 45. * u.deg
     az = 0. * u.deg
     rot_tel = Angle(0, unit= u.deg)

     target_name="Rotator test"
     time_data = await mtptg.tel_timeAndDate.next(flush=True, timeout=2)
     curr_time_mtptg = Time(time_data.tai, format="mjd", scale="tai")
     time_err = curr_time_mtptg - Time.now()
     print(f"Time error={time_err.sec:0.2f} sec")

     # Compute RA/Dec for commanded az/el
     cmd_elaz = AltAz(alt=alt, az=az,
                      obstime=curr_time_mtptg.tai,
                      location=location)
```

Test Cycle Execution

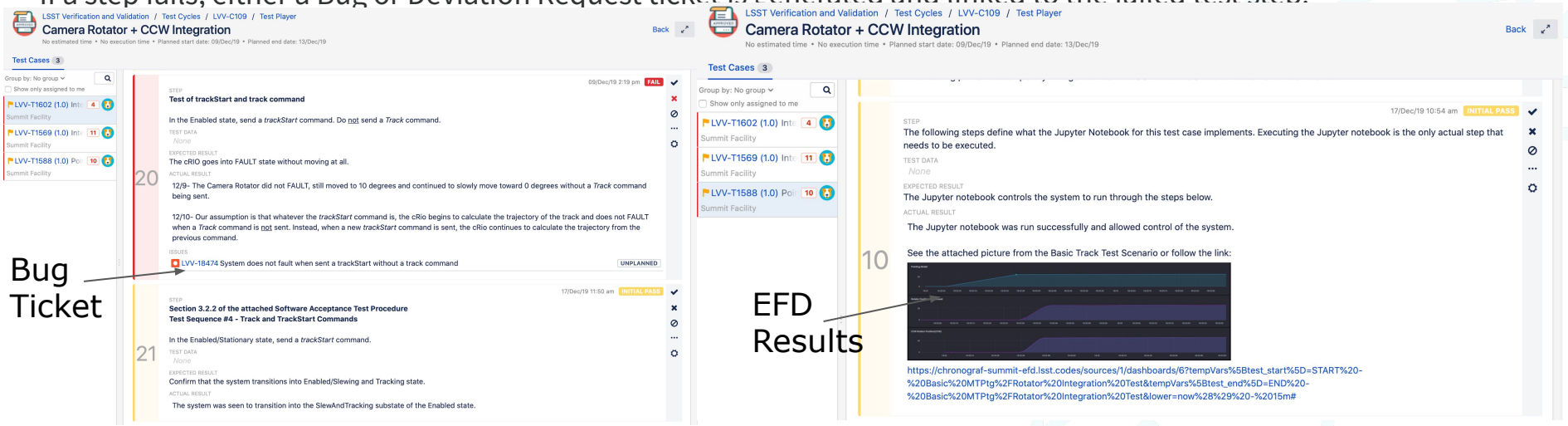
During execution of a Test Cycle, each Test Case is loaded up in sequence

The user populate the Actual Result field is along with setting the status of each step.

The Jupyter Notebook is executed by the user and results are captured in the notebook and EFD.

The EFD is queried via the Chronograf interface or the notebook to analyze the results as required.

If a step fails, either a Bug or Deviation Request ticket is generated and linked to the failed test step.



Verification Report Generation

Test Plan/ Report PDF includes results from the Test Execution, Bug / Deviation Request Tickets, and Overall Assessment and Recommended Improvements fields of the Test Plan.

4 Test Campaign Overview

4.1 Summary

Test Plan **LVV-P58: CCW + Camera Rotator Interface Verification on Camera Cart**			Completed
Test Cycle **LVV-C109: Camera Rotator + CCW Integration**			Done
test case	**status**	**comment**	**issues**
LVV-T1602	Fail	Overall, the test campaign for the integration of the Camera Rotator with SAL 4.0 was seen as a FAIL. We observed an issue with the *trackStart* command where the history of the previous *trackStart* command needed to be cleared in order for the system to calculate the trajectory for a new command. Otherwise, the system would continue to calculate the original command and allow for a demand to be given that was out of acceptable range.	LVV-18471 LVV-18474 LVV-18474 LVV-18474
LVV-T1569	Fail	Overall, the test campaign for the CCW + HR interlock integration was a FAIL. We found that we were unable to synchronously move the CCW and Camera Rotator with just a *move* command and instead had to use a *track* command. Then when the synchronous motion limit switches were tested, both	LVV-18473

Status: **Fail**

50 Description

Hard Breaking E-Stop Test

The following steps define what the Jupyter Notebook for this test case implements. Executing the Jupyter notebook is the only actual step that needs to be executed.

Expected Result

The Jupyter notebook controls the system to run through the steps below.

Actual Result

We successfully ran the Jupyter notebook which allowed proper control of the system.

For the EFD plot of the final run of the E-STop Test, see the attached picture or follow the link:

https://chronograf-summit-efd.lsst.codes/sources/1/dashboards/6?tempVars%5Btest_start%5D=START%20-%20E-Stop%20MTPtg_Rotator_CCW%20Integration%20Test&tempVars%5Btest_end%5D=END%20-%20E-Stop%20MTPtg_Rotator_CCW%20Integration%20Test&lower=now%28%29%20-%2015m#

Status: **Initial Pass**

Verification Elements Development

For vendor developed systems that were verified by the vendor previously, we review the requirements and determine the subset of functional requirements that need to be re-verified and generate a new Verification Element.

In the Verification Ticket in Jira we define specifically what needs to be verified by populating the following fields.

- Verification Method
- Success Criteria
- Description
- Requires Monitoring?

Optional

- Verification Phase
- Verification Level