

# ENGINEERING **METHODS FOR VERIFICATION &** VALIDATION

Holger Drass, Austin Roberts for the V&V team













## **Overview**

- 1. Verification System Architecture
- 2. <u>Testing at the Summit</u>
- 3. Backup Material



2



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

3



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.



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

5



## **Verification System Architecture**

6



## **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** C RUBIN FRVATORY Jira Software à wedieel am Syndeia Requirements **Deviation Tickets** <<sync>> **Verification Tickets Test Case** Execution Result Focus will be on the right side. A Test Case is used Test Cycle for all verification methods **Test Manager** for Jira



## 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) 0
  - Mapping to Verification Phases (Test Cycles) 0
  - Definition of Verification Procedures (Test Cases) 0
  - 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 0 determination on the sufficiency of results
  - Tracks verification progress 0
  - Monitors verification compliance after 0 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.

LSST Verification and Validation / LVV-12030 LTS-218-REQ-0053-V-01: 3.16 SAFETY PULL CORD_1				LSST Verification and Validation / Test Cases / LVV-T1569 (1.0) Integrated CCW + HR Interlock Test Save New	w Version 1.0 ····
Sedit Q Comme	nt Assign More - Descoped Submit for Rev	iew Admin 🗸		Details Test Script Execution Traceability Attachments Comments History	
Details					
Туре:	E Verification	Status:	IN VERIFICATION (View Workflow)		+ 0 -
Priority:	★ Undefined	Resolution:	Unresolved	Couerane	
Component/s:	T&S			CVERage	IN VERIFICATION
Labels:	None				
Requirement Details	Verification Details			~ Confluence	Q
Verification Method:	: Test			No confluence pages	
Success Criteria:	The limit switches get tripped at +/- 2.2 deg. The C	CW and Camera Rotato	r transition to Fault State and stop motion.		
Percentage of Test	0.0%			Veb Links	Q
Passing:				No web links	
Description					
Verify that when the 0	CCW and Camera Rotator are out of sync by more than po	sitive 2.2 degrees and ne	egative 2.2 degrees the limit switches are tripped and		
stops the motion of b	oth the CCW and Camera Rotator.				
		/			
Traceability			+ -		
Test Cases					
Coverage					
> LVV-T1569 (1.0) Ir	ntegrated CCW + HR Interlock Test				
					1



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)



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



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**

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 Plan Review Verification Elements to determine what requirements to verify 3. Populate Verification Elements details 4. 5. Trace Verification Elements to Verification Procedures (Test Case in Jira) 6. Organize and sequence Test Cases into **Test Cycles** Develop details and steps of Test Cases 7. Implement Create Jupyter notebook to perform the Test Case 8. 9. Produce Test Plan / Test Report document Schedule event in JIRA summit calendar and procure equipment, personnel, etc. 10. 11. Execute Test Cases and capture execution results in Jira Follow up on the findings by creating any Deviation Requests, FRACAS reportable 12. **Execute** failures, and Bug tickets 13. Produce updated Test Plan / Test Report with results, assessment, and recommendations

Vera C. Rubin Observatory | PCW 2023



## **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





- 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**



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

Let's go a small tour.

Let's go a small tour. Actually we have two ways to fill a test in our JIRA test manager Zephyr:	st case execution $ \begin{array}{c} A test Case vs Test \\ time you execute a template. Execution.\\ an existing one viction or go on est Case, vo \end{array} $
Zephyr – Jupiter Notebook/ Love SALScript – EFD – Zephyr	Zephyr – Block ticket – Love SALScript – EFD – <sup>Sas</sup> vou have as Interpreter – Zephyr
jupyter 🛞	

A test case is a template. Each







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

← → C A ≠ <sup>2</sup> https://jiu.kstcorp.org/browse/8L0CK-34	යා ලද∦ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	← → C lb love01.cp.lsst.org/uit/view?id=44	ත් (C)	9 y 10 0 0 0 🗧 🖉 🕈 🐠 🕸 🖒 =
JITA Software Dashboards - Projects - Issues - Boards - Calendar Tests Plvot Reports - Scrum Standup Create	Q. Search 🥐 🕜 🔅 👰		10:28:01	ب 🖕 🗢
Gateway Test for the M1M3 Dynamic Test, low speed, low acceleration - 5% motion settings		мтолеле		raw data
III Q Comment Assign More - Propose Reobserve	🖾 Email < Pivot Report 🍵 Export 🛩	STATE	CURRENT SCRIPT	
_ ≥ Details Lα Type: C Block Status: MonSteel Motify; ~ High Resolution: Done	✓ People Assignee: Sruno Quint Assign to me	Summary State ENABLED (* Queue State RUNNING ()	None Elipped time: 0.0 s Estimate time: 0.0 s	_
Componente: Marine Que Labeir: None Componente: None Componente: Componente: Componente	Beporter:  Bruno Quint Votes:  O Vote for this issue Watchens:  Start watching this issue		Contiguring Script: track_target 40 × 500	✓ Show details ▼
This test is formuly described in UV-27884.081-Gatesay Test for the MMM Dynamic Test, bus speed, flow acceleration - Data Catestion, You can not it by following the steps before: © • Set up the TMA.EU with 5% motions settings	<ul> <li>♥ Dates</li> <li>Creared: 27(July(23 2:17 PM Updated)</li> <li>10(July(23 7:21 PM Resolved: 10(July(23 6:00 PM</li> </ul>	AVAILABLE SCRIPTS (87) Filter:	<ul> <li>description: Differential tracking rate in RA (sec/sec).</li> <li>type: number</li> <li>ype: object</li> <li>type: notice</li> </ul>	
az vel 8.5 az cel 8.5 az jent 3.4 el vel 8.5 el 1.5 el 1.5 zel 1.6	* Development           12cmmbh           6.commbh           1.ouf reseast           1.ouf reseast           Updated 03(Jul/23 4:50 PM	> mini	39 Junget: 40 dditionalProperties: false 41-ascription: Optional configuration section. Find a target to p 42 the given position and magnitude range. If not specified, the 43 "operties:	er S
<ul> <li>Move the treescope up and down in elevation with long seles using the transarter (right) institute (move_p.g., py solpt and the computation below:</li> <li>as 1 133</li> </ul>	✓ Jenkins No builds found.	√ SCHEDULER	<ul> <li>44 dz:</li> <li>45 description: Azimuth (in degrees) to find a target.</li> <li>46 type: number</li> </ul>	Ð
e1: [34, 22, 34] puise_for: 5 progress: NLOCF-34 reason: SIXTORF-82	<ul> <li>× Agile</li> <li>View on Board</li> <li>✓ Calendar events</li> </ul>	/enable.py	MANUAL CONFIG         FORM CONFIG         Select an option         •           1 # Insert your config here:         •<	8
Last_color made: UV-72834 execution: UV-7277 vergino 1.0	No events     Issue is not scheduled in Calendar yet	/load_snapshot.py	2 # e.g.; 3 # wait_time; 3600 4 # fail_run: false	
ann, iomail 2003 Ippore: ['nesas', 'nesasta', 'nesasta', 'nesastajetiay', 'nesa', 'nebaupod_i', 'nebaupod_i', 'nebau']	Schedule Issue Create calendar	/standby.py	5 # fail_cleanup: false	
		ATOUFUE		raw data
		STATE	CURRENT SCRIPT	
		Summary State ENABLED (8)	None Elapsed time: 0.0 s	



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.

FK5

import logging
import yaml
import os
import sqlite3
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
<pre>import astropy.units as u</pre>
from astropy.time import Time
from astropy.coordinates import AltAz, ICRS, EarthLocation, Angle
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.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 mtpg.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(("Time error={time\_err.sec:0.275 sec")



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.

No estimated time + No execution time + Planned start date: 09/Dec/19 + Planned end date: 13/Dec/19			No estimated time * No execution time * Planned start date: 09/Dec/19 * Planned end date: 13/Dec/19				
est Cases 3			Test Cases 3				
p by: No group v Q how only assigned to me		STEP 09/Dec/19 2:19 pm FAL  STEP 2:19 pm FAL  ST	Group by: No group 🗸				
VV-T1620 (1.0) Int: 4 3 mit Facility VV-T1589 (1.0) Int: 11 3 mit Facility VV-T1588 (1.0) Po: 10 3 mit Facility	20	In the Enabled state, send a trackStart command In the Enabled state, send a trackStart command. Do not send a Track command. In the Enabled state, send a trackStart command. Do not send a Track command. In the Enabled state, send a trackStart command. Do not send a Track command. In the Enabled state, send a trackStart command is a track command. In the Enabled state, send a trackStart command is a track command. In the Enabled state, send a trackStart command is a track command. In the Enabled state, send a trackStart command is a track command. In the State state without moving at all. In the State state without is that whatever the trackStart command is, the cRio begins to calculate the trajectory of the track and does not FAULT In the State state state state state with an uncerkStart command is is send the CRI company from the track and does not FAULT In the State	Strep       Strep       17/Dec/19 10:54 an ID         Summit Facility       The following steps define what the Jupyter Notebook for this test case implements. Executing the Jupyter notebook is the only actual st needs to be executed.         Test Purchase       Test Purchase         Summit Facility       Expected Beautr         Summit Facility       The Jupyter notebook controls the system to run through the steps below.         Actional RESULT       The Jupyter notebook was run successfully and allowed control of the system	NITIAL PASS × step that 0 			
Bug Ficket	21	previous command. UNPLANNED UV-18474 System does not fault when sent a trackStart without a track command UNPLANNED TODOLV-18474 System does not fault when sent a trackStart without a track command TODOLV-18474 System does not fault when sent a trackStart command. Test Sequence #4 - Track and TrackStart Commands In the Enabled/Stationary state, send a trackStart command. Test Dia. None Common that the system transitions into Enabled/Slewing and Tracking state. Confirm that the system transitions into Enabled/Slewing and Tracking state.	EFD       10         See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: See the attached picture from the Basic Track Test Scenario or follow the link:         Image: State Test Scenario or follow the link: <tr< td=""><td></td></tr<>				
		The system was seen to transition into the SlewAndTracking substate of the Enabled state.					

Camera Rotator + CCW Integration

ra Rotator + CCW Integration

Back 🚽



## **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			
Test Cycle LVV-C109: Camera Rotator + CCW Integration			Done
test case	status	comment	issues
		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 com-	
LVV-T1602	Fail	mand needed to be cleared in order for the sys-	LVV-18471
		tem to calculate the trajectory for a new command.	LVV-18474
		Otherwise, the system would continue to calculate	LVV-18474
		the original command and allow for a demand to	LVV-18474
		be given that was out of acceptable range.	
		Overall, the test campaign for the CCW + HR inter-	
		lock integration was a FAIL. We found that we were	
		unable to synchronously move the CCW and Cam-	
		era Rotator with just a move command and instead	
		had to use a track command. Then when the syn-	
LVV-T1569	Fail	chronous motion limit switches were tested, both	LVV-18473

#### 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



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?

### <u>Optional</u>

- Verification Phase
- Verification Level