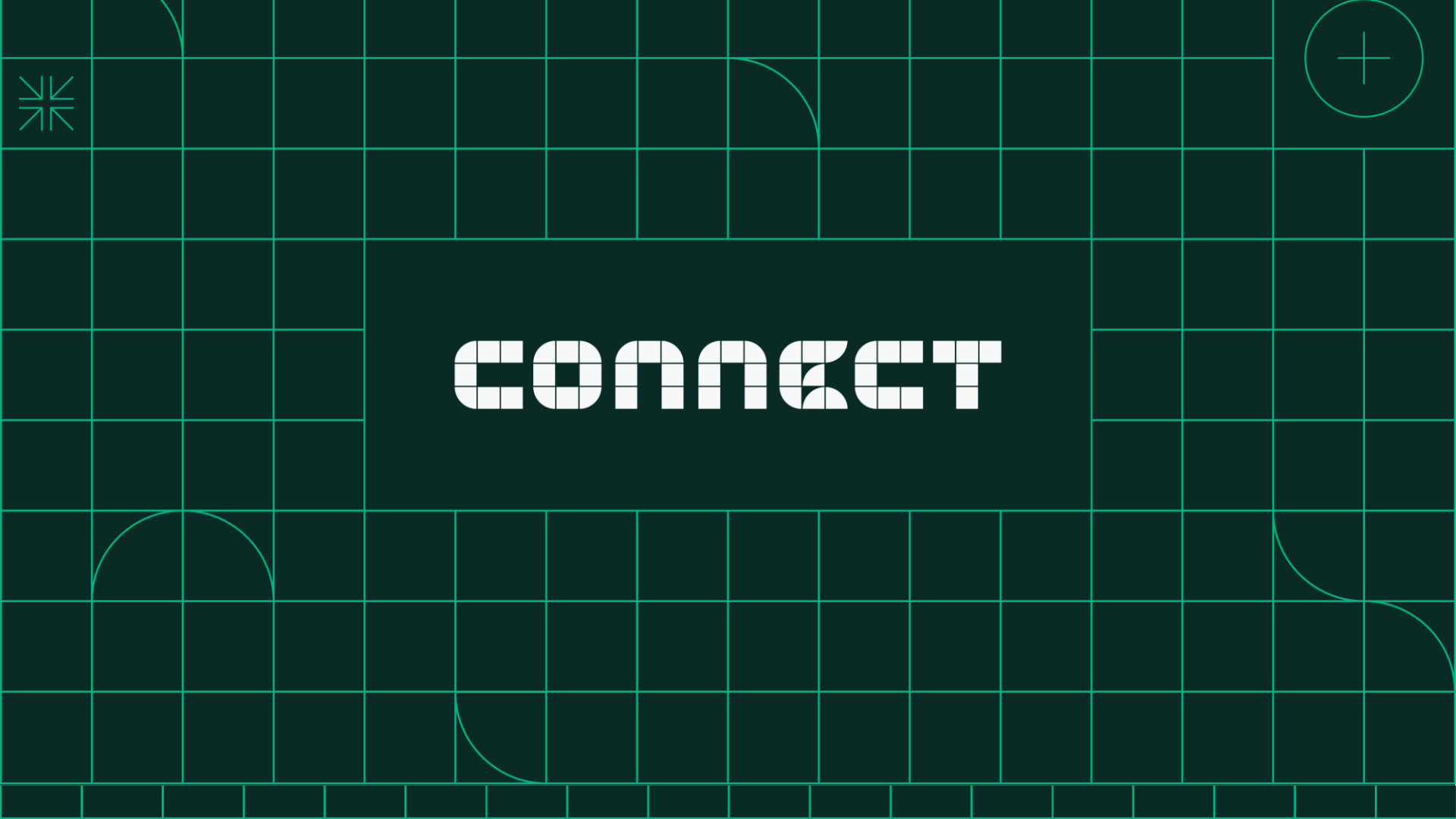


W  **LC**  **ME**  **AUST**  **N**

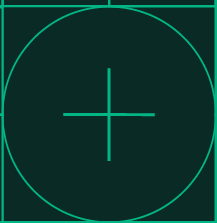


CONNECT



CONNECT

2023 AUSTIN

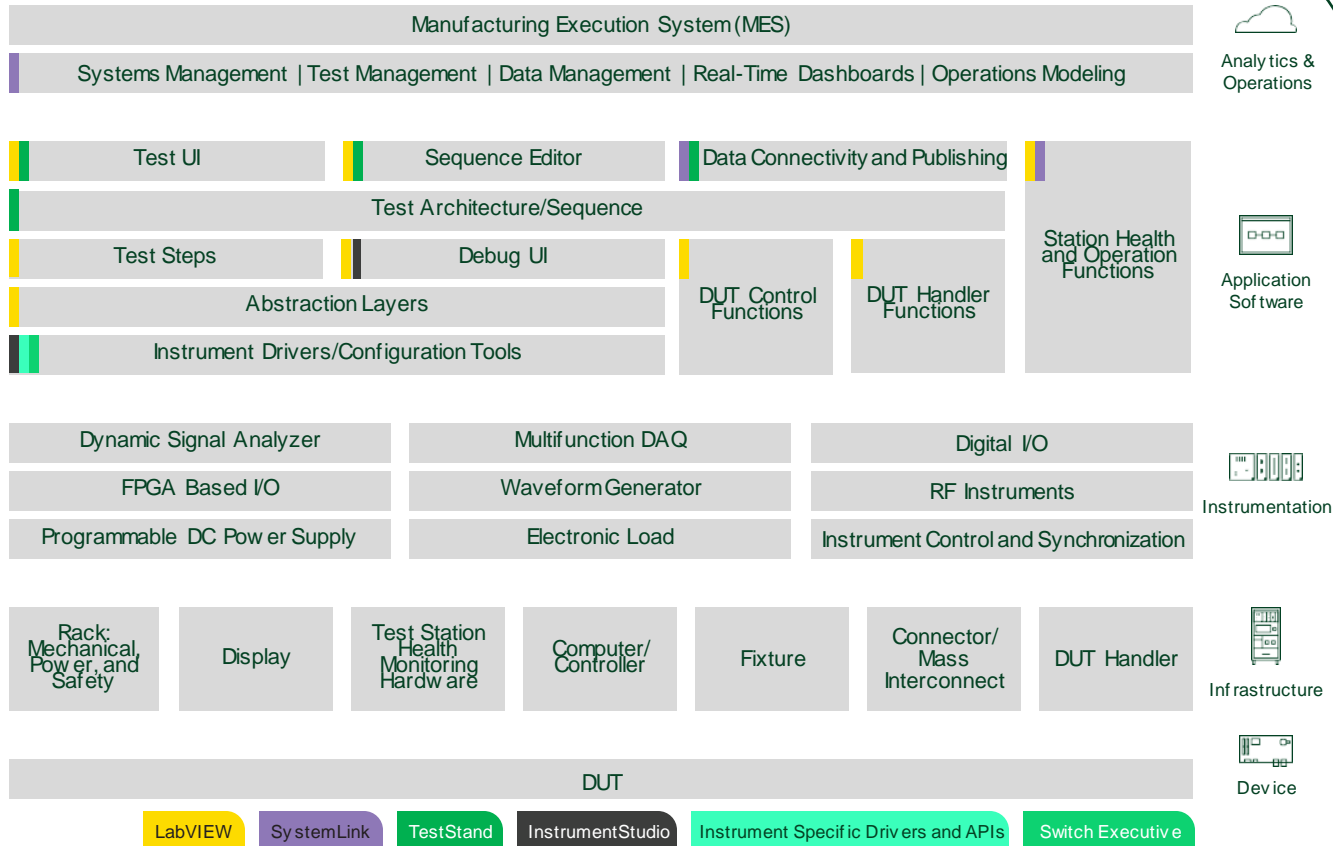


Continuous Integration with LabVIEW and TestStand

Nathan Nims - Digitalisation
Business Manager, NI

Callum Eggleton – Test Engineering
Design Manager, Thales UK

Elements of a Test Architecture



Operation, Maintenance, and Regulatory Documentation



Analytics & Operations



Application Software



Instrumentation



Infrastructure

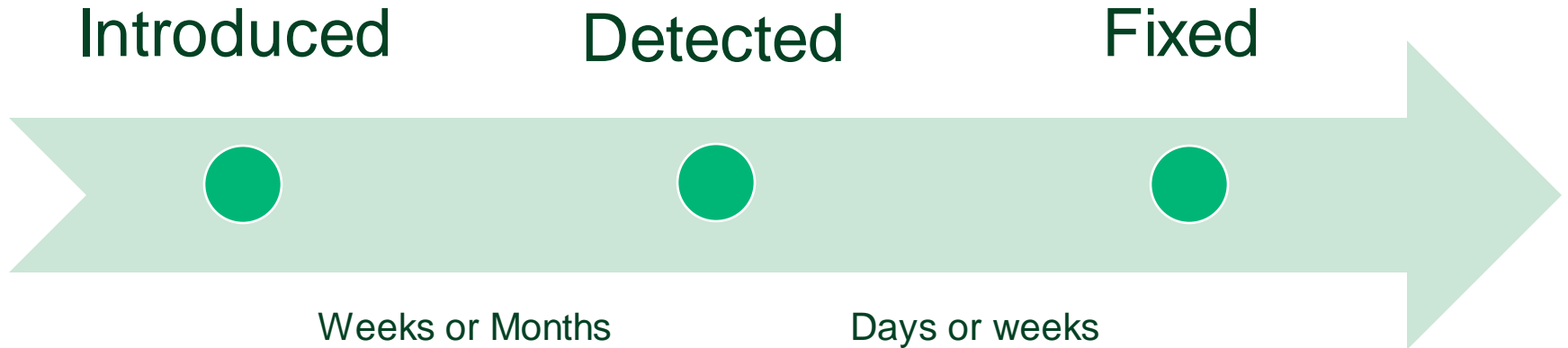


Device

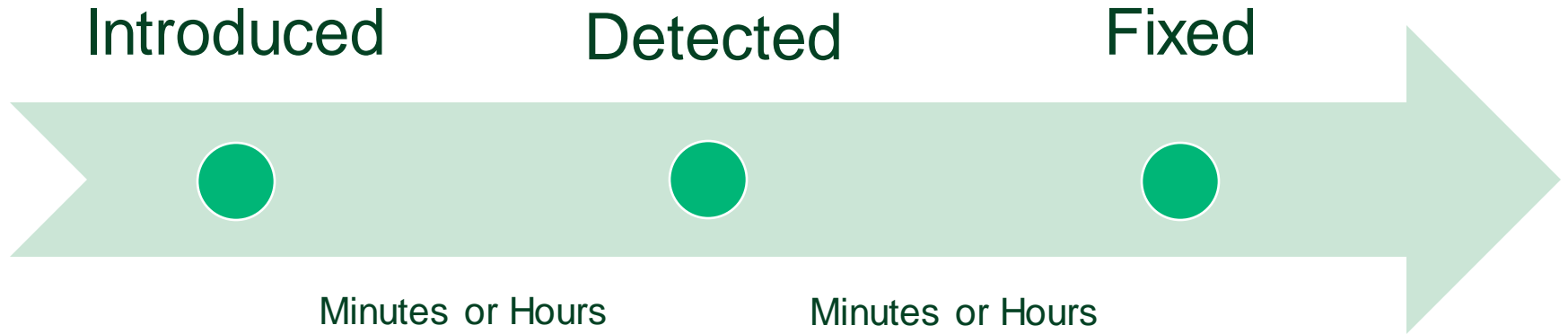
Bug lifecycle

Where do things go wrong?

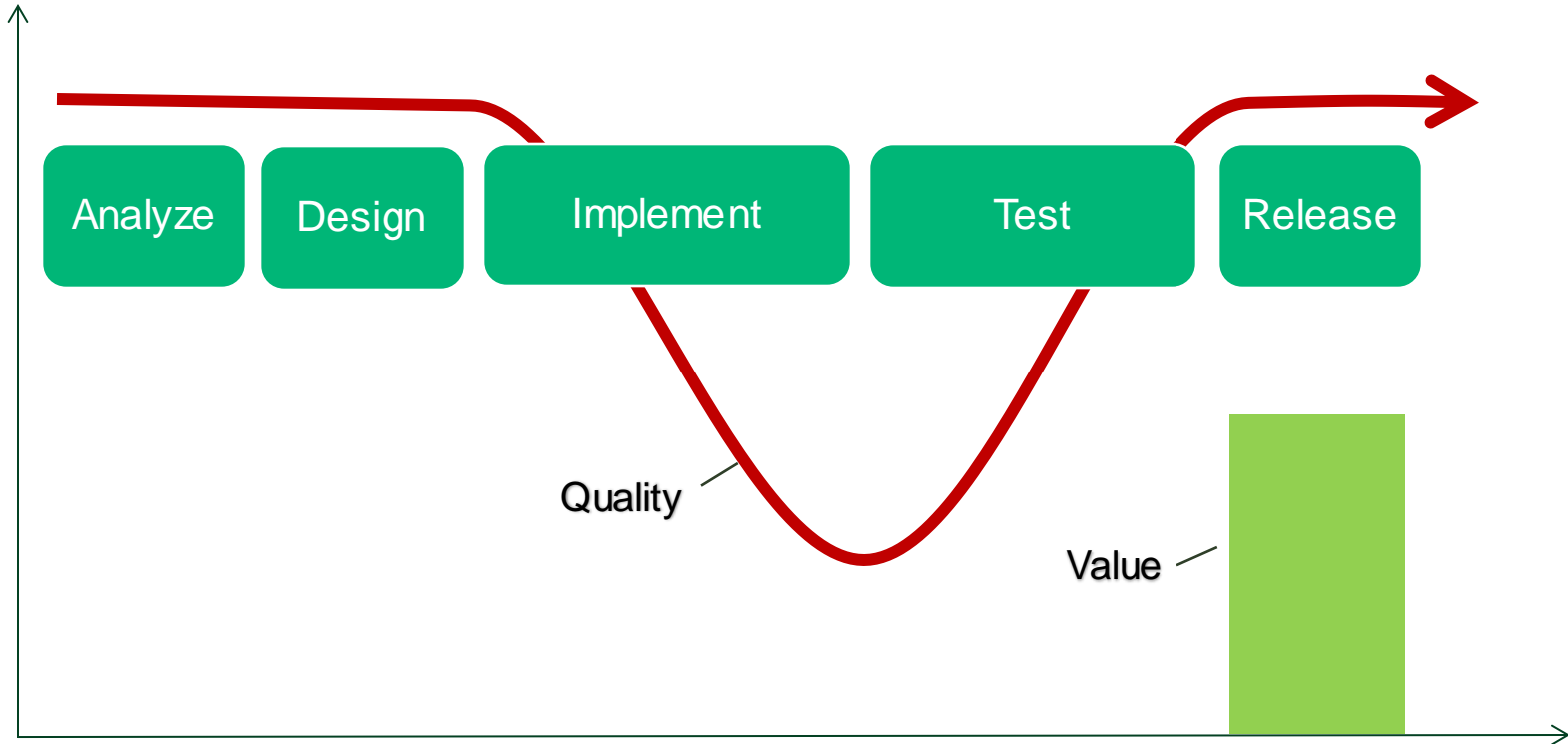
- Waiting a long time between code checks
- Don't run the checks thoroughly every time
- Don't bother to write good tests
- Don't look through the test/analysis results to see what broke
- Humans have a limited attention span
- Repetitive testing is boring and usually skipped by humans



ni Bug lifecycle – After CI

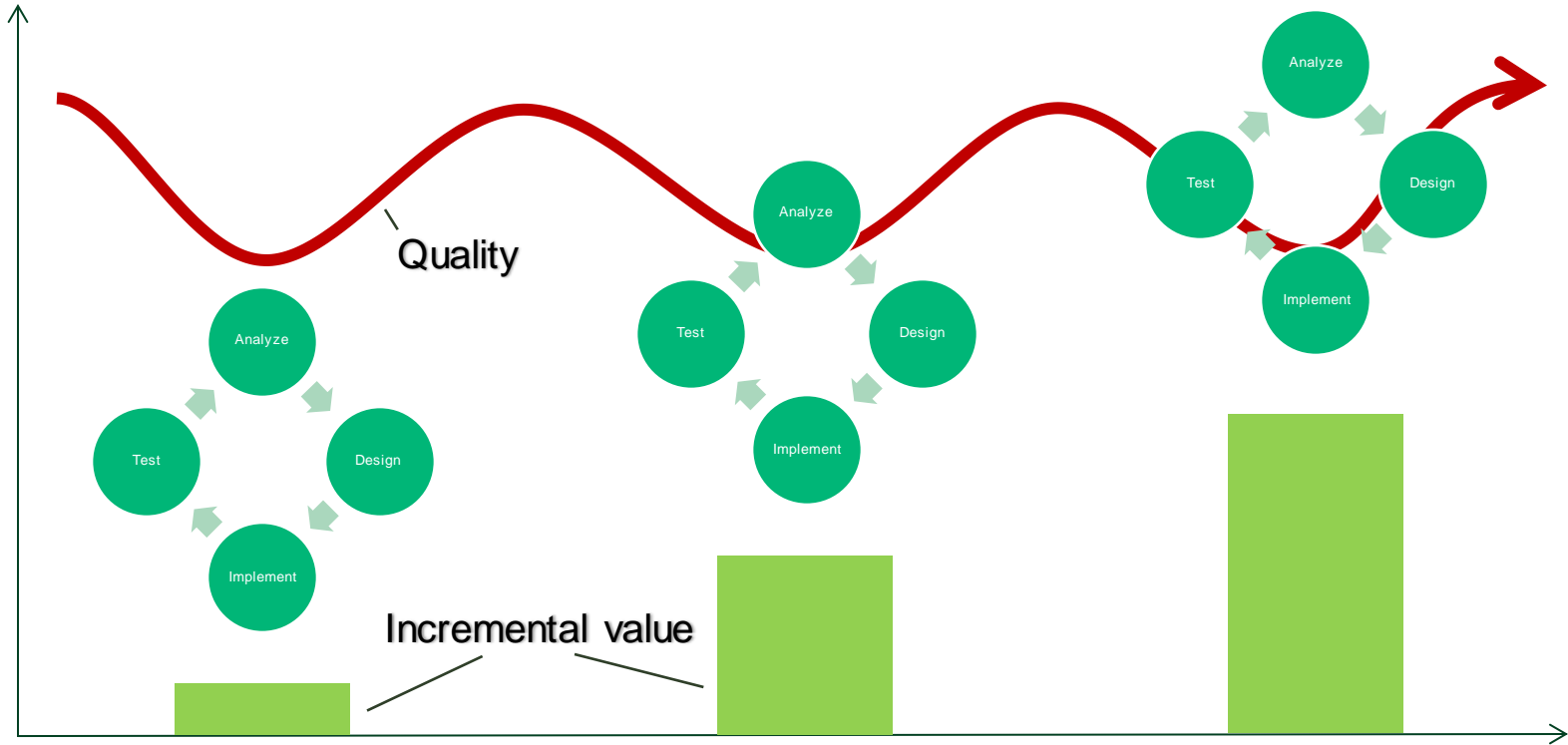


ni Traditional Waterfall Development



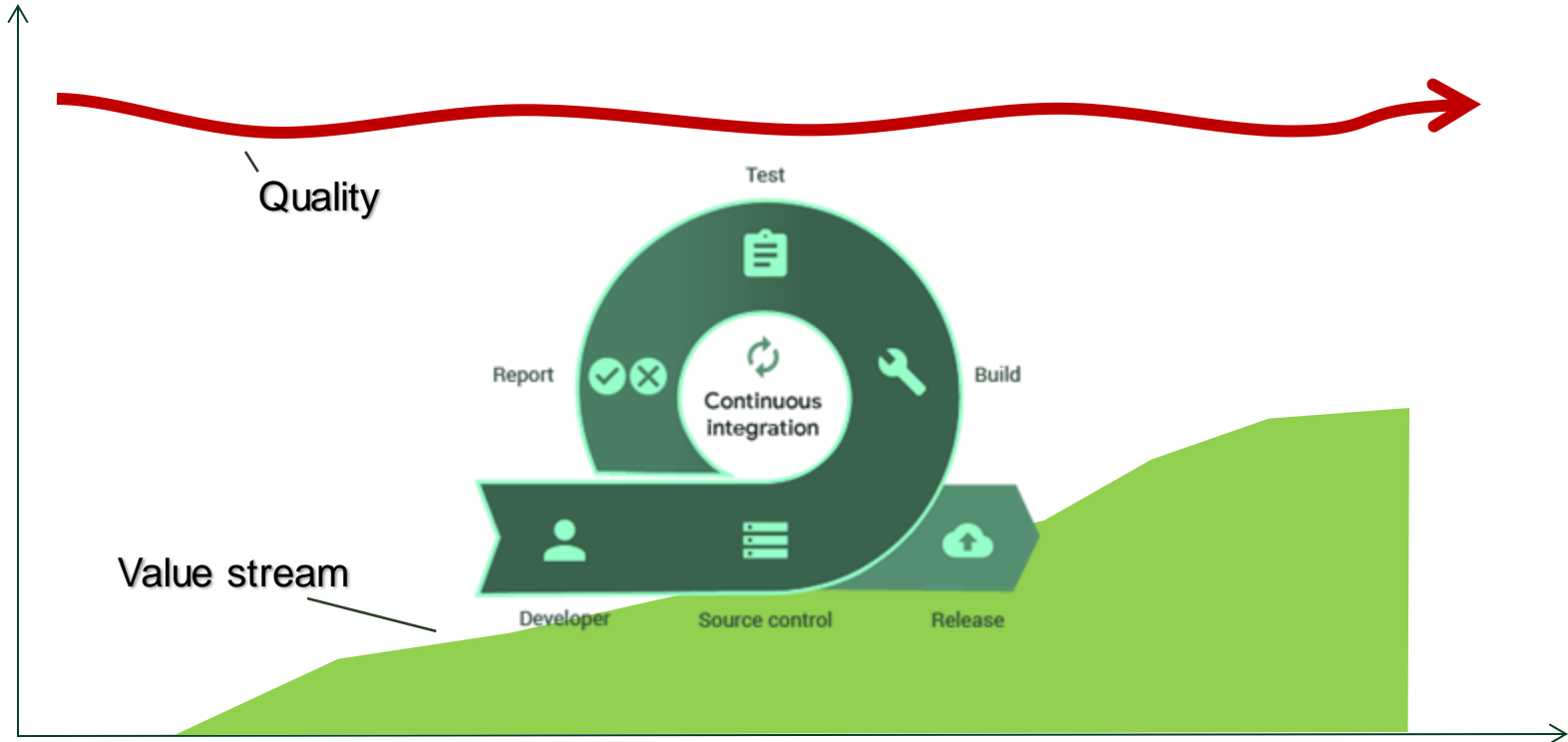


Agile





Continuous Integration Continuous Deployment (idealized)



THALES

Building a future we can all trust



Continuous Integration with LabVIEW and TestStand

Test Engineering at Thales UK

Callum Eggleston – Test Design Engineering Manager

www.thalesgroup.com

OPEN



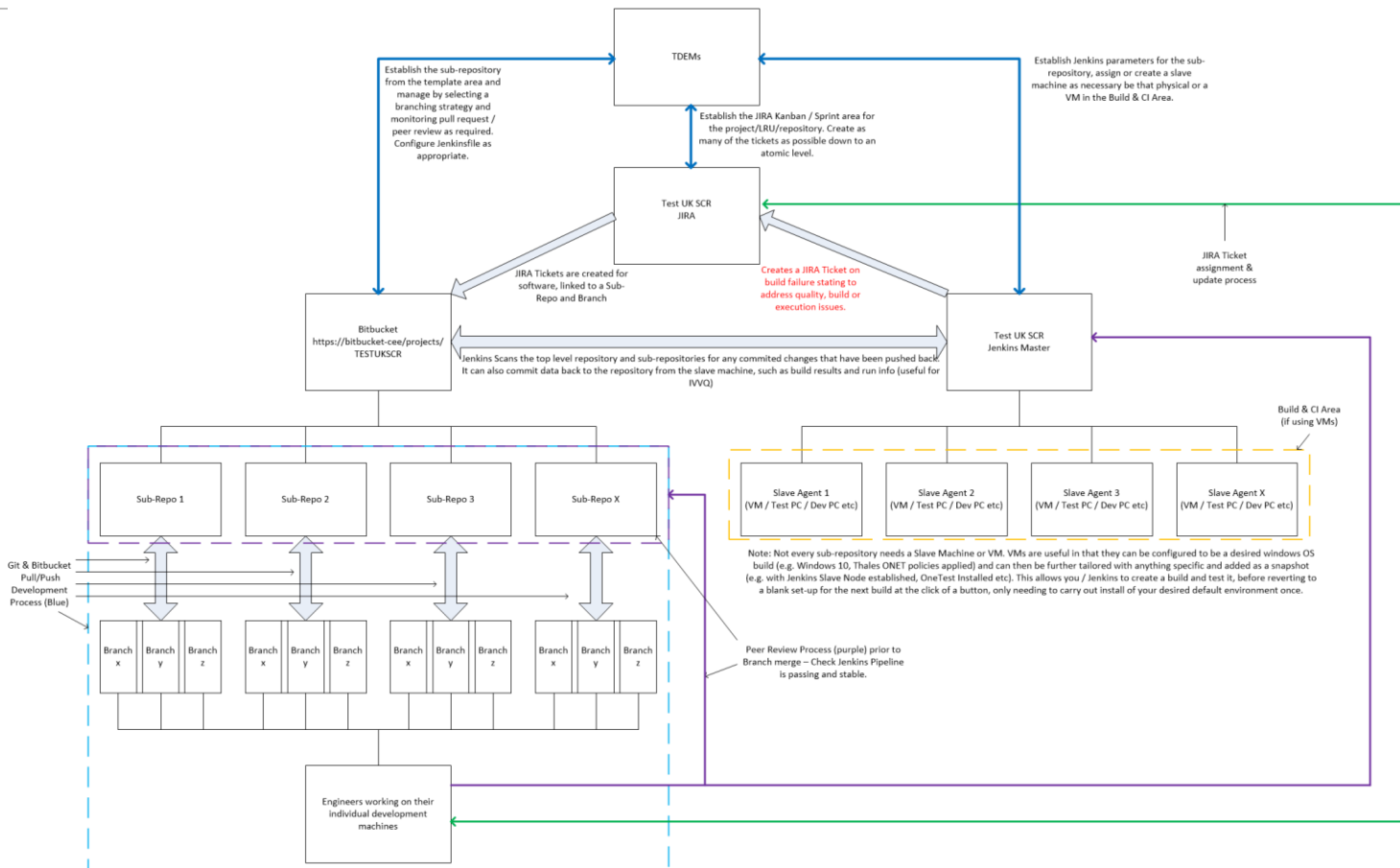
What is a Pipeline?

- When triggered, an orchestrator (build automation server) will execute a staged set of actions (the pipeline) on your target source code and publish the results back to the orchestrator.
- This set of actions is user defined and the tasks to be executed can be tailored for each branch of each project or project phase that wishes to use a pipeline.
- The ability to trigger automatic execution of these tasks on detection of a SW change makes pipelines perfect for CI tasks. Carrying out static analysis, Unit testing, build checks and even testing out different configurations with a HIL setup on every change made.



Tools & Process Overview

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.



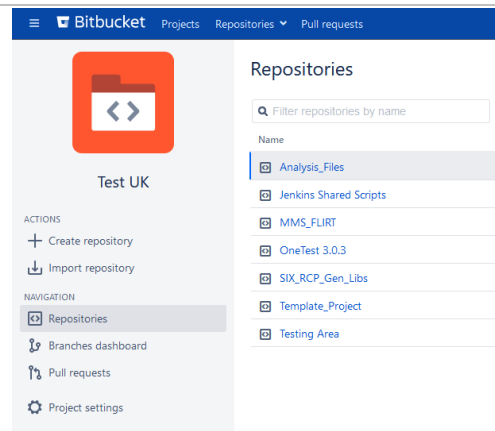
Test SW Framework – Source Control (Bitbucket & Git)

This is our SW project area. All repositories are kept here, including common departmental code and project specific code.

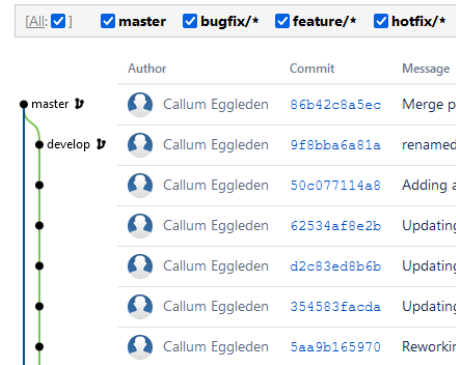
Git based system, develop locally and store remotely – each commit is tracked and can be rolled back to when required.

Each repository can have different levels of user access – useful for controlling security for those who have/haven't signed project SALs.

Multiple Engineers can work on the same code at once using Branches. If SW is designed and managed properly (JIRA), merge conflicts can be minimized.



All Branches Graph



Lilly Pipe	4106df9508	Merge pull request #39 in TOPSIS-X/systems-ivvq from feature/TOPSISX-6386 to develop * commit
Callum Eggleston	8795c82b469	TOPSISX-6386 Resolve Conflicts
Callum Eggleston	1f8858c8b10	TOPSISX-6386 Fixing CCP VI to work with CCP Type Enum
Callum Eggleston	b29cedbdc1	TOPSISX-6386 Resolving Merge Conflicts
Callum Eggleston	b1236eca293	Merge pull request #38 in TOPSIS-X/systems-ivvq from feature/TOPSISX-6336 to develop * commit
Lilly Pipe	86a6af6019a	TOPSISX-6386 Calling correct Type Defs
Callum Eggleston	b88ab1e8d21	TOPSISX-6336 Updated Tiger CCP Status FGV and added a new utility to find uncalled VIs within th
Callum Eggleston	8aad553e9ec	TOPSISX-6386 trial VI
Callum Eggleston	8f333b8c12e	TOPSISX-6336 Tiger Typedefs updated to be of correct type in order to work with CCP general VI

This document may not be reproduced, modified, adapted, published, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.

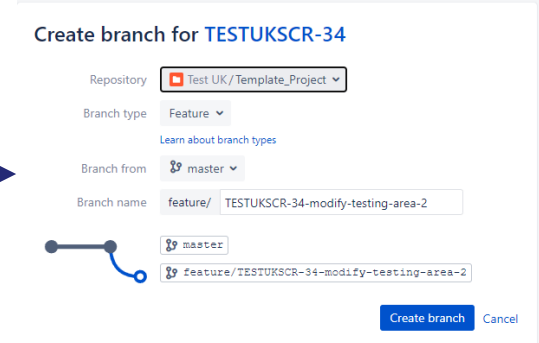
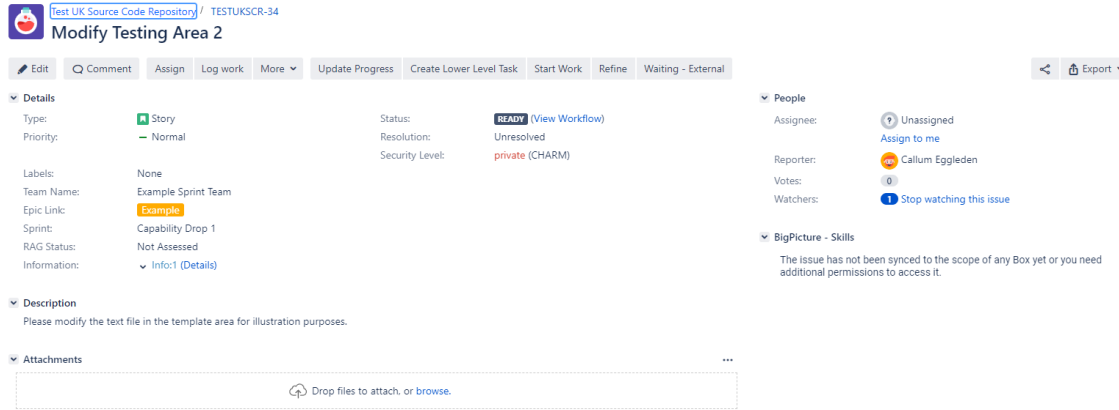
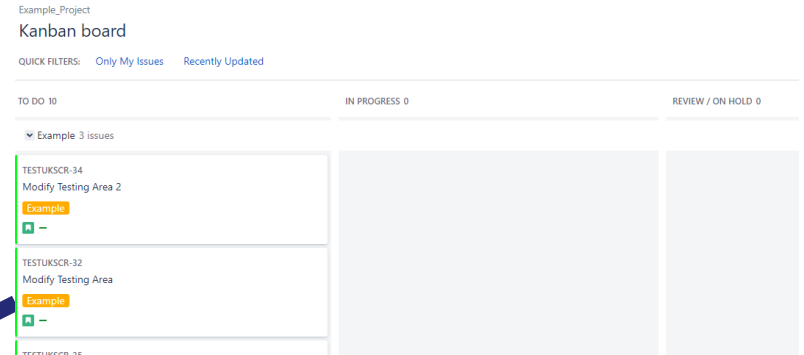
Test SW Framework – SW Management (JIRA & Bitbucket)

This is our Project Management area in JIRA.

Create Kanban boards / sprints to show the status of tickets for individual projects, using a naming convention to filter.

Create ticket detail with data from project requirements (HW & SW) and internal SW design documentation.

Create Bitbucket Branches from Tickets to manage SW Development.



OPEN

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.

Test SW Framework – Jenkins

Jenkins is a Build Automation Server. This instance of it is the project area for Test UK at Thales.

It is linked to our Test UK project in Bitbucket & scans all repositories in there for a “Jenkinsfile” – A text file with this exact name & no extension.

This Jenkinsfile is set up by the user and defines our CI Pipeline for that instance/branch of the project.

Any repositories with a Jenkinsfile are presented in the project view. Clicking the repository will open up Jenkins view of it and present all active branches.

A slave agent is set up by Jenkins and created on the target machine, usually a VM or test machine connected to hardware.

Results dashboard detailing a history of each pipeline execution and changes in quality, errors etc.

The screenshot shows the Jenkins interface for the 'Test UK' project. On the left, there is a navigation menu with options like 'Up', 'Status', 'Configure', and 'Scan Organization Folder Log'. The main area displays the 'Test UK' project details, including the folder name 'TUK' and a list of repositories. One repository, 'template_project', is highlighted with a red circle. Below this, the 'Stage View' table provides a detailed breakdown of pipeline stages and their execution times for two different builds.

	Declarative: Checkout SCM	Unit Tests	VI Analyzer	Sequence Analyzer	Build	Execute	Report	Requirements Analysis	Store	Declarative: Post Actions
Average stage times:	4s	41s	30s	31s	20s	15s	15s	13s	1s	970ms
#11 Jun 22 14:41	2s	38s	40s	34s	9s	22s	10s	13s	2s	979ms
#10 Jun 22 11:14	5s	25s	27s	22s	6s	18s	16s	11s	2s	971ms

CI Pipeline Tools – Jenkinsfile, Groovyscripts

“Jenkinsfile” – This is a text file that sits in the main area of a repository with this exact name & no extension, that defines our CI Pipeline.

It tells Jenkins which Build Executor (Agent / Slave) to use. Each stage is defined and environment variables setup.

The Jenkinsfile can be configured to switch stages in or out, create new stages etc. We have standard pipelines but these can be configured per project and/or on maturity of project.

The Jenkinsfile calls “Groovyscripts”. These are reusable functions, kept in a separate repository (library) that Jenkins calls without needing to deploy to the target. The Jenkins Slave on the target machine executes the Jenkinsfile & Groovyscripts.

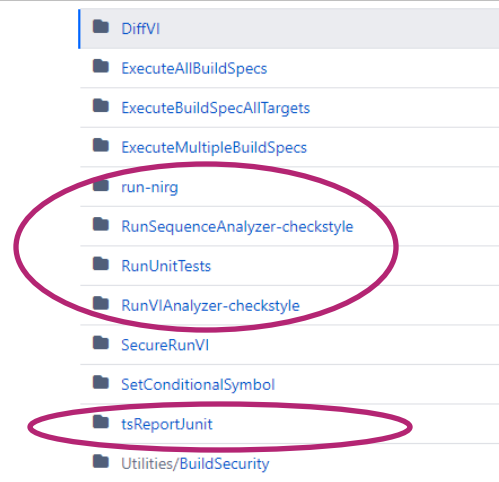
We keep the groovyscripts as a repository in bitbucket that Jenkins calls as a shared library. This means the groovyscripts are under source control and aren't deployed with each project, preventing accidental changes, uncontrolled scripts etc.

```
*C:\Projects\Project_Name\Jenkinsfile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Jenkinsfile VIAtest.groovy runNrg.groovy
1 pipeline {
2   agent { label 'Test_UK_SCR_Framework' }
3   stages {
4     stage('Unit Tests') { //Please see Test Wiki for Groovyscript description, their functions and
5       steps {
6         script{
7           env.labviewPath_2021 = "C:\\Program Files (x86)\\National Instruments\\LabVIEW 2021
8           env.SeqAnalyzer = "C:\\Program Files (x86)\\National Instruments\\TestStand 2021
9           env.teststandPath = "C:\\Program Files (x86)\\National Instruments\\TestStand 20
10        }
11       utfTest("Test_Code\\Project_Name.lvproj")
12      }
13     }
14     stage('VI Analyzer') {
15       steps {
16         bat "git clone ssh://git@bitbucket-cee:7999/testukscr/analysis_files.git C://Jenkins
17         VIAtest("C:\\Jenkins\\analysis_Files\\Development.viancfg","Test_Code",10)
18       }
19     }
20     stage('Sequence Analyzer') {
21       steps {
22         bat "xcopy /e /y /i \\${WORKSPACE}\\Test_Sequences\\" "C:\\Jenkins\\Builds\\Test_Seq
23         SeqAtest("C:\\Jenkins\\analysis_Files\\Development.tsaproj",10)
24       }
25     }
26   }
27 }
```

```
C:\Projects\jenkins-shared-scripts\vars\VIAtest.groovy - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Jenkinsfile VIAtest.groovy runNrg.groovy
1 def call(path, targetDirectory, threshold){
2   echo 'Run static analysis before build'
3   catchError(buildResult: 'SUCCESS', stageResult: 'SUCCESS') {
4     bat "LabVIEWCLI -OperationName RunVIAnalyzer-checkstyle -ConfigPath \
5   }
6   recordIssues enabledForFailure: true, qualityGates: [[threshold: "${thres
```

CI Pipeline Tools – LabVIEW CLI / CMD

- Jenkins can use command line commands. These are either in the Jenkinsfile or part of a Groovyscript (set function).
- Some commands are standard CMD prompts for directory changing and file manipulation, others use LabVIEW CLI. LabVIEW CLI must be installed to drive LabVIEW to control the NI Suite of tools that forms our pipeline.
- Most LV CLI commands come as standard but we have created custom ones which we store in a separate repository. These need to be deployed to target PCs / VM before the pipeline can operate properly.
- The image on the right shows both Standard & Custom (circled) LV CLI functions.
- The image below this shows how it is called inside a Groovyscript.
- Without support from NI to create these & tie the rest of our framework together, we wouldn't have a CI pipeline. We estimate a saving of 6 months to setup.



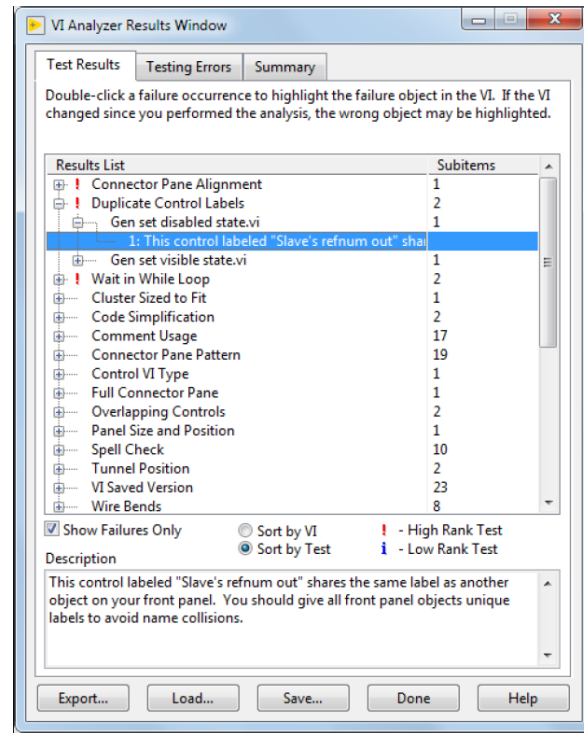
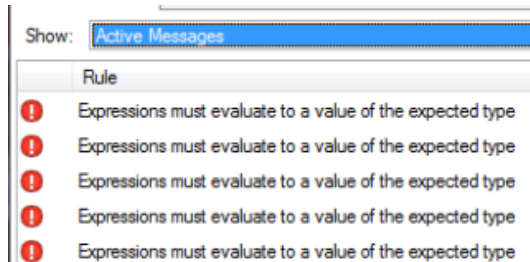
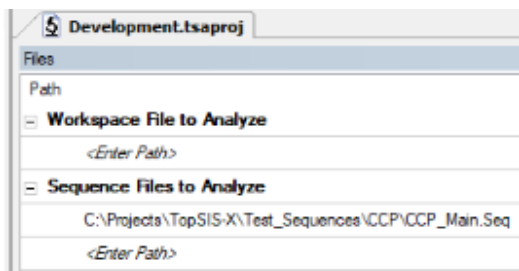
```
Jenkinsfile x VIATest.groovy x runNirg.groovy x
1 def call(path){
2   echo 'Run Requirements Gateway'
3   bat "LabVIEWCLI -OperationName run-nirg \"%${path}\" \"nirg_result"
4   echo 'JUnit'
5   junit allowEmptyResults: true, testResults: "nirg_results.xml"
6 }
```

CI Pipeline Tools – Static Analysis (VI Analyzer & Sequence Analyzer)

These are the tools we use to check the VIs & sequences we create to ensure they meet our coding standards. This increases bug detection, maintainability and re-use. Currently performs 75 checks per VI and over 10 checks per sequence step.

The config files for both analyzers are kept in a separate repository, which is deployed to the target machine and automatically used by the CI Pipeline, it does not require any user training or interaction post kick-off.

The pipeline is configured to target a LabVIEW project and Sequence file(s) / workspace for analysis through the Jenkinsfile.



CI Pipeline Tools – SW Testing (Unit Test Framework)

Unit Test Framework – allows repeated functional validation as part of the pipeline. A test is created for each relevant VI. This could be; safety critical test VIs, VIs tied directly to a requirement, VIs simulating the functionality of a product or even VIs that get heavily re-used (HAL).

The CI Pipeline targets the project file and executes all unit tests within, executing the tests every time the pipeline is run.

This is ideal for regression testing & can be used to automatically generate validation documents.

The screenshot shows the 'Unit Test Framework Results' window with the following data:

Test Results		Test Errors	
Passed	1	.Ivtest File	0
Failed	0	VI under Test	0
Skipped	0	Setup VI	0
Error	0	Teardown VI	0
Total	1	Test Execution	0
Total		Total	0

Statistics		VIs	
Total Time	00:00.71	VIs in Project	206
Project Code Coverage	71.4%	VIs Passed	1
		VIs Failed	0
		VIs Indirectly Tested	0
		VIs Not Tested	205

The screenshot shows the 'Test Properties : Sample.Ivtest' window. It displays test cases and their configurations. The 'Test Cases' section shows a diagram of a test case with inputs and outputs. Below it, there are two tables:

Input Name	Data Type	Input Value
Setup VI		
VI under Test		
<input checked="" type="checkbox"/> Input	Double Float	3
<input checked="" type="checkbox"/> Condition	Enum U16	0
<input checked="" type="checkbox"/> error in (no error)	Cluster	FALSE
<input checked="" type="checkbox"/> status	Boolean	FALSE
<input checked="" type="checkbox"/> code	I32	0
<input checked="" type="checkbox"/> source	String	

Output Name	Data Type	Comparison	Expected Value
VI under Test			
<input checked="" type="checkbox"/> Output	Double Float	=	6
<input checked="" type="checkbox"/> error out	Cluster	=	FALSE
<input checked="" type="checkbox"/> status	Boolean		FALSE
<input checked="" type="checkbox"/> code	I32		0
<input checked="" type="checkbox"/> source	String		
Teardown VI			

The screenshot shows the 'Project Explorer' window for the 'TopSIS-X.lvproj' project. The file structure includes:

- Analysis_Files
- Artifacts
- Test Sequences
- TopSIS-X_CCP_&_User_Node
- CCP_Front_Panel
- Common
- GenericVIs
- Messages
- TestVIs
 - Lamp_Test
 - Masking_Test.vi
 - Sample.Ivtest
 - UN_Msg_Response_Test_Driver.vi
 - Write_To_CCP.Ivtest
 - Write_To_CCP.vi
- User_Node
- User Node and CCP Structure.png
- .gitignore
- Jenkinsfile
- TopSIS-X.aliases
- TopSIS-X.lvps
- Dependencies
- Build Specifications

CI Pipeline Tools – Test Execution

By setting up a physical PC as a Jenkins slave agent and installing the required tools (LV / TS / Analysis tools etc), we schedule builds of our CI pipeline connected to actual test hardware and UUT.

Allowing us to execute full tests.

With proper error handling, this allows for a majority of longer tests to be run reliably overnight/weekends.

Different configurations of simulators are setup in different Bitbucket branches using config files. This allows for easy R&R, regression, integration testing. All scheduled in Jenkins.

Test results overview displayed starting with any failures, then all differences highlighted. We use a custom LV CLI step to parse the TestStand results report to into Junit format to allow for this.

All Failed Tests

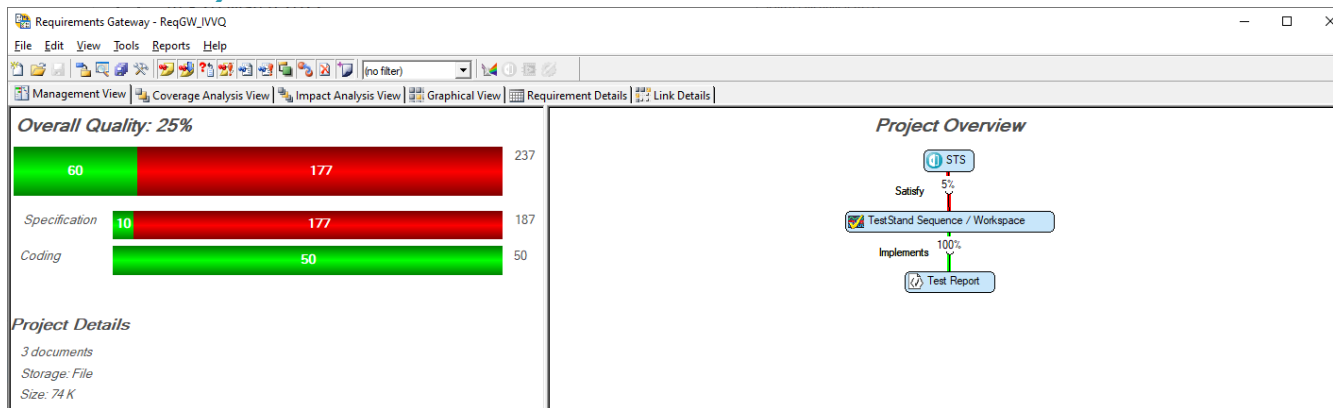
Test Name
Report / Module_1.seq\CPU Test\Register Test..Test_Step
Error Details
Data Name: Register Test Details: Start Date / Time : 2023-05-04 @ 14:35:22.023
End Date / Time : 2023-05-04 @ 14:35:22.023
Step Type : PassFailTest
Step Group : Main
Test Time : 3.93e-05
Test Data : Name = PassFail ---- Value = TRUE
Requirements Analysis / IVVQ.rqtfl.Requirement\PAS_PRJ.Requirement
Requirements Analysis / IVVQ.rqtfl.Requirement\Output Voltages Diagnostics.Requirement
Requirements Analysis / IVVQ.rqtfl.Requirement\Power Supply Diagnostics.Requirement
Requirements Analysis / IVVQ.rqtfl.Requirement\Input Voltages Diagnostics.Requirement
Requirements Analysis / IVVQ.rqtfl.Requirement\Advanced Power Management Diagnostics.Requirement

All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
IVVQ.rqtfl.Requirement\	0 ms	5 -10	0	46 +10	51
Module_1.seq\CPU Diagnostics\	0 ms	0	0	5 +5	5 +5
Module_1.seq\CPU Test\	0 ms	1 +1	0	4 -1	5
Module_1.seq\Main Sequence\	0 ms	0	6 +4	5	11 +4
Module_1.seq\Power Supply Test\	0 ms	0	0	3	3

CI Pipeline Tools – Requirement Analysis (Requirements Gateway)

- Allows linking between requirements documentation (DOORs/PAS) <> Test Code (Sequences / VIs) <> output reports of an execution of that test code. Automates tracking of progress through a project so far.
- Shows coverage between each link, i.e. how much of the requirements are covered in the sequence? How much of that sequence was executed in the last run? Therefore showing how many requirements were last met in that SW run.
- Bitbucket branches can contain SW configured to test all aspects in situations where test code may not execute a requirement in ideal conditions. E.g. a diagnostics subsequence may not execute when every test passes, but that subsequence must still exist, still be called on a failure and still be tested properly to ensure requirements are met.
- Only needs to be configured at project kick-off or when entering a new development phase (e.g. transitioning from IVV to Production).



Review Process – Bitbucket & Jenkins

When an engineer initiates a pull request to merge a branch back into develop, the reviewer is presented with an overview of the changes.

The changes are immediately clear from commit comments & the fact that the build failed shows there are issues present.

Following circled link takes you through to the Jenkins Dashboard (BlueOcean in this case)

Callum Egglesten develop → master MERGED

Develop

Overview Diff Commits

Details

Callum Egglesten created a pull request 03 Jan 2023

- Updating sequence to point to VIs from the PPL created in the build step
- updating project to build to a set path in Ciprojects/builds so the PPL can be found by the sequence
- Updating sequence file to use relative path for the PPL

1 build failed Learn more

Activity

What do you want to say?

Callum Egglesten MERGED develop to master in commit 830e8e06c21 09 Jan 2023

Martin Holloway marked the pull request as APPROVED 03 Jan 2023

Callum Egglesten OPENED the pull request 03 Jan 2023

Review Process – Bitbucket & Jenkins

This document may not be reproduced, modified, adapted, published, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.



! TestUK_Dev / template_project 1 > Pipeline Changes Tests 1 Artifacts Logout X

Branch: develop [x] 11m 17s No changes

Commit: b91d013 4 months ago Branch indexing

X 1 tests have failed

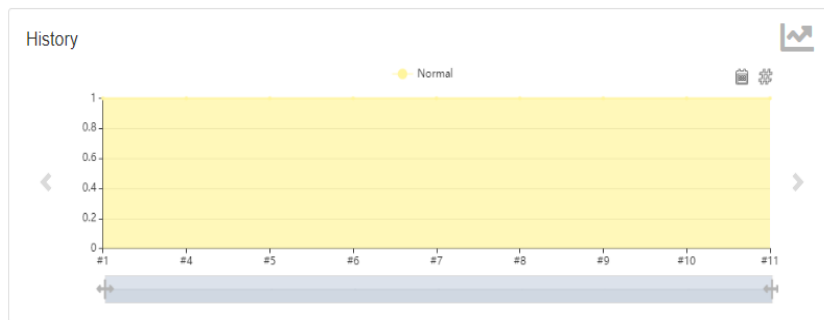
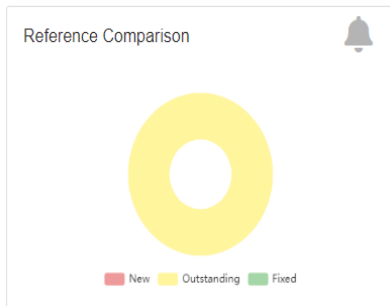
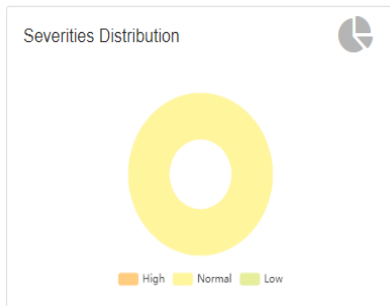
There are 0 new tests failing, 1 existing failing and 2 skipped.

Existing failures - 1

X > Unit Tests / Test Case 1 - Project_Name\Ivproj\My Computer\Unit Tests\Replace_me.Ivtest <1s

CI Pipeline Tools – Jenkins Dashboard View (VI Analyzer)

VI Analyzer Warnings

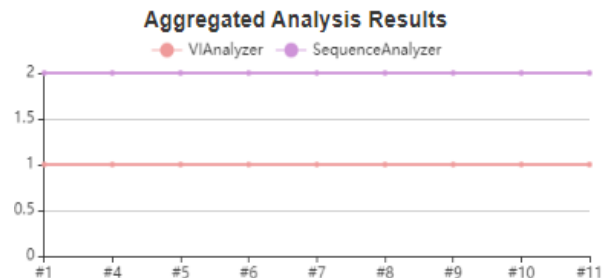


Issues

Show 10 entries

Details	File
	Replace_me2.vi:0

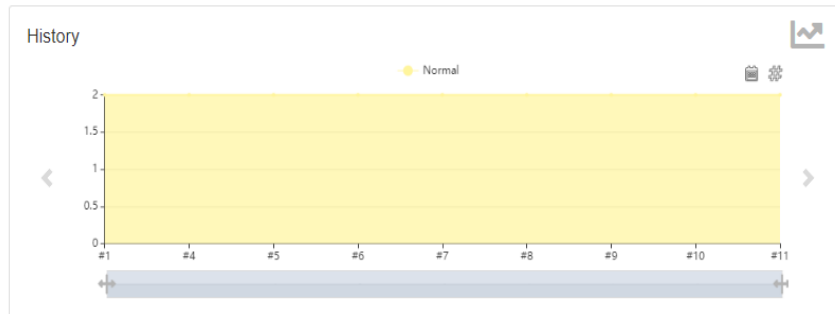
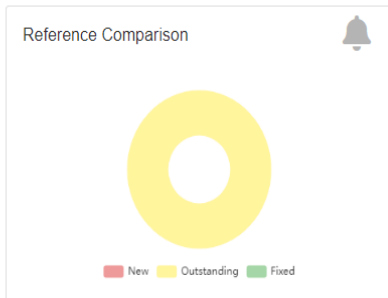
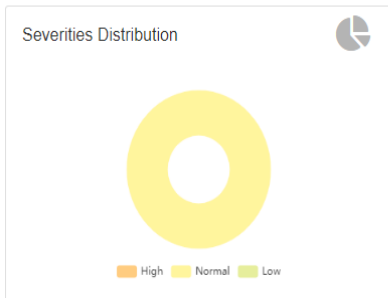
This block diagram contains fewer than 1 comment.



This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.

CI Pipeline Tools – Jenkins Dashboard View (Sequence Analyzer)

This document may not be reproduced, modified, adapted, published, distributed, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.



Details

Issues

Show 10 entries

Details

File

Module_1.seq:0

Step run mode is set to Skip. in Data.Seq["MainSequence"].Setup["Simulation Dialog"].T.S.Mode

CI Pipeline Tools – Junit Test Result Detail

Test Result

17 failures (±0)

All Failed Tests

Test Name
+ Unit Tests / Project_Name\proj\Unit_Tests\..Test Case 1
+ Unit Tests / Project_Name\proj\Unit_Tests\..Test Case 2
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\CPU Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\ROM Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\RAM Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Video Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Keyboard Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Else..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Power Supply Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Register Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Instruction Set Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Cache Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\FPU Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Speed Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Input Voltages Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Output Voltages Diagnostics..Requirement
+ Requirements Analysis / ReqGW_Prod.rqt\Requirement\Advanced Power Management Diagnostics..Requirement

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
Project_Name\proj\Unit_Tests\	52 ms	2	0	0	2
ReqGW_Prod.rqt\Requirement\	0 ms	15	0	38	53

Test Result

4 failures (+4), 16 skipped (+16)

40 tests (+20)

Took 3 min 43 sec.

[add description](#)

All Failed Tests

Test Name	Duration	Age
- CHROME_48_0_2564_97_Linux..guineaPig_1.Guinea Pig Assert Title 1 - A		
- Error Details		
Testing if the page title equals "I am a page title - Sauce Labs - A".	4.3 sec	1
+ Stack Trace		
- FIREFOX_45_0_LINUX..guineaPig_1.Guinea Pig Assert Title 1 - A		
- Error Details		
Testing if the page title equals "I am a page title - Sauce Labs - A".	7.1 sec	1
+ Stack Trace		
+ INTERNET EXPLORER_11_WINDOWS..guineaPig_1.Guinea Pig Assert Title 1 - A	7.9 sec	1
+ MICROSOFTEDGE_undefined_ANY..guineaPig_1.Guinea Pig Assert Title 1 - A	16 sec	1

All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
CHROME_48_0_2564_97_Linux..	24 sec	1 +1	4 +4	5	10 +5

This document may not be reproduced, modified, adapted, published, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.

Summary of Benefits

- **Automated regression testing - SW Unit testing** ensures the detection of unexpected changes & behaviour in existing code which can be fixed before it becomes a bigger issue – saves a large number of wasted hours. This testing occurs on every push back to the repository and every merge.
- **Automated requirements capture** - Links requirements from test sequences back to DOORs / PAS documents for IVVQ / Production. Captures project status and any scope creep - will aid in understanding & estimating of future projects.
- **Automated quality checks** - Enforces coding standards on VIs & Sequences, making it easier for multiple engineers to work on and understand.
- **“Passive training”** where people will naturally adjust their code to meet coding standards prior to pipeline feedback.
- **Improved code quality & bug detection** – Better, clearer code makes it easier, maintain understand and re-use. Faster bug detection allows for quick fixes while in the same headspace.
- **Can execute fully automated tests** in either VMs or Physical Machines & report back (useful for HIL or Gauge R&R).
- **A Dashboard of pipeline results** - Reduced time spent in reviews, quality status available “at a glance”. Results from each build are kept together with the individual build so it’s all tied together.



demo [TestUK_Dev = template_p] x

https://xbnlci1451.ukdn.thalesuk/job/TUK/job/template_project/job/d...

BitBucket Jira Wiki Jenkins Build & CI Area TOL Thales Systems Benefits Other favourites

Jenkins

1 search

Eggleden, Callum | log out

Jenkins > TestUK_Dev > template_project > demo

ENABLE AUTO-REFRESH

- Up
- Status
- Changes
- Build Now
- View Configuration
- Full Stage View
- Open Blue Ocean
- Bitbucket
- VI Analyzer Warnings
- Sequence Analyzer Warnings
- Embeddable Build Status
- Pipeline Syntax

Build History

trend

#301	15-May-2023 10:19
#300	15-May-2023 10:10
#299	15-May-2023 10:05
#298	10-May-2023 09:09
#297	10-May-2023 09:01
#296	04-May-2023 14:30
#295	21-Apr-2023 14:03
#294	21-Apr-2023 14:03
#293	21-Apr-2023 14:03
#292	21-Apr-2023 14:03
#291	21-Apr-2023 14:03

Questions

This document may not be reproduced, modified, adapted, published, translated, in any way, in whole or in part or disclosed to a third party without the prior written consent of THALES - © 2023 THALES. All rights reserved.

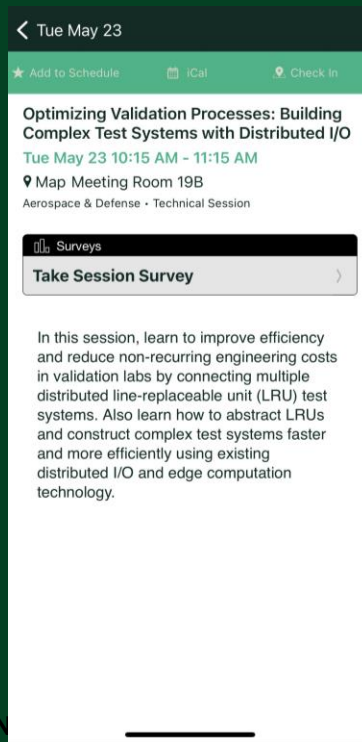
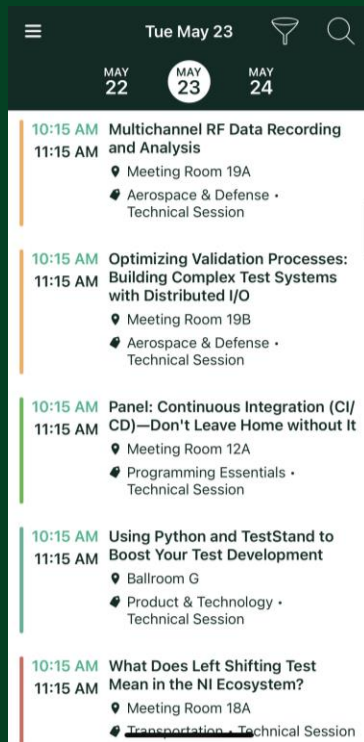
Links

- Continuous Integration – Learning Path
 - <https://learn.ni.com/learning-paths/continuous-integration>
- VI Analyzer – checkstyle
 - <https://github.com/LabVIEW-DCAF/VIA-Checkstyle>
 - <https://forums.ni.com/t5/Continuous-Integration/VI-Analyzer-Results-gt-Checkstyle-Format/td-p/3741067?profile.language=en>
- Requirements Gateway - JUnit
 - <https://forums.ni.com/t5/Distributed-Control-Automation/Requirements-Gateway-in-Continuous-Integration/td-p/3915133?profile.language=en>

Give us your feedback!

Quick 2 Question Survey

In the mobile app,
click into the
session you would
like to provide
feedback for



Click "Take the
Session Survey"