NI CONNECT

2024 AUSTIN

EMERSON

EMERSON | NI

Optimizing Test System Development: Best Practices Unveiled

CONNECT

- Raul Galvez
  - 16-year veteran in the test automation industry specializing in the design, development, and deployment of automated test systems (ATE's) for various types of customers across multiple countries. Certified LabVIEW and TestStand architect as well as a python practitioner with more than a decade mentoring customers and peers ranging from architects to developers. Former employee of Sanmina-SCI (Contract manufacturer), Averna (NI alliance Partner), and currently part of National Instruments (NI) as a Chief Solutions Engineer.

ARCHITECT ARCHITECT

PCEP™ – Certified Entry-Level Python Programmer
PI
OpenEDG Python Institute

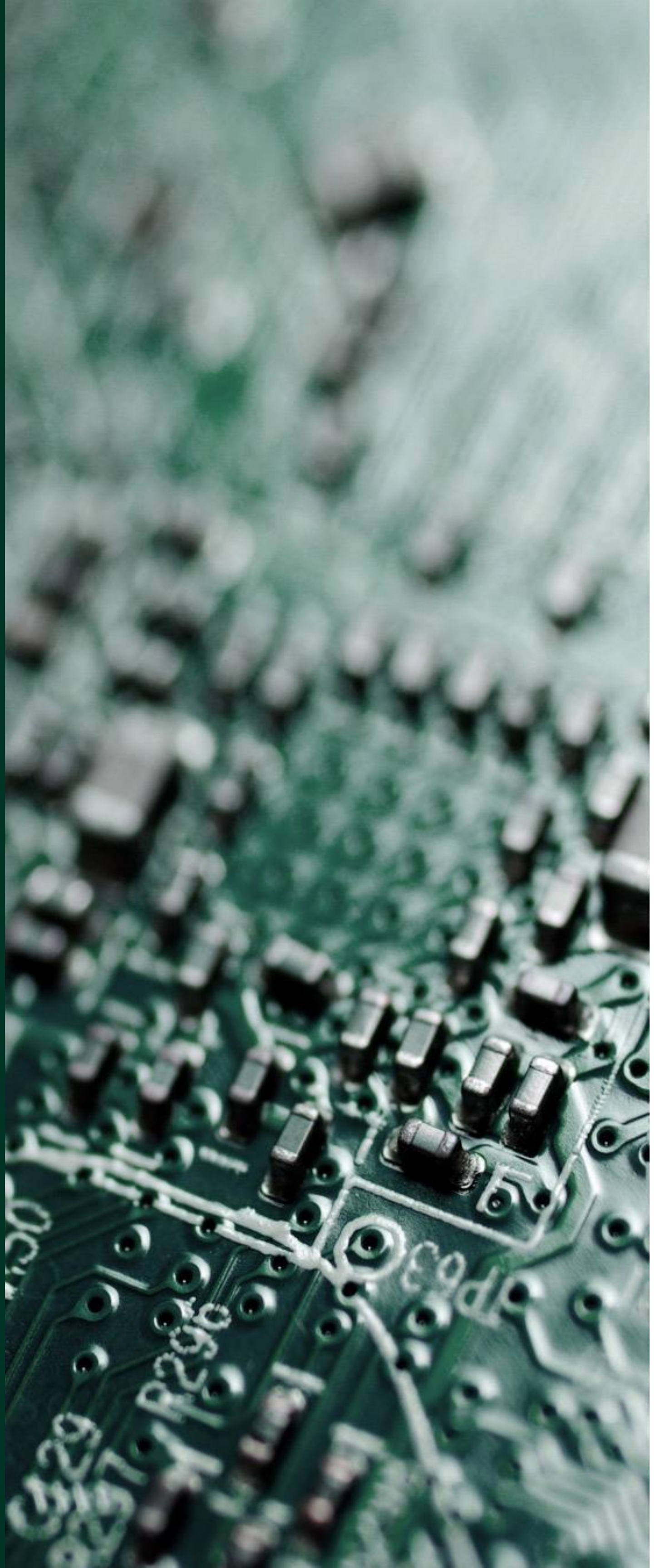
- Sergio Velderrain
  - Mechatronic engineer with 12 years of experience designing solutions using NI software. Former employee of multiple NI partners (Cygnus, Averna, and Konrad). LabVIEW Champion before joining National Instruments and a Certified LabVIEW and TestStand architect. ADAS subject matter expert and founding member of the first ADAS datalogging team for Konrad Germany. Currently part of NI as a Senior Field Applications Engineer for automotive customers in the northern California area.

ARCHITECT ARCHITECT

LabVIEW™ CHAMPION

- Adolfo Islas
  - Electronic Engineer/MBA with 11 Years of Experience as a test system integrator and Reckon Solution commercial manager. Designed and deployed projects for multiple industries ranging from automotive, consumer electronics and telecommunications in Mexico, Canada, USA, Argentina and Brazil. Passionate LabVIEW and TestStand developer currently contributing to the NI community with LabVIEW user groups over social media with over 6.5K users. Content creator and streamer of Developing LabVIEW practices in Twitch and Facebook Live.

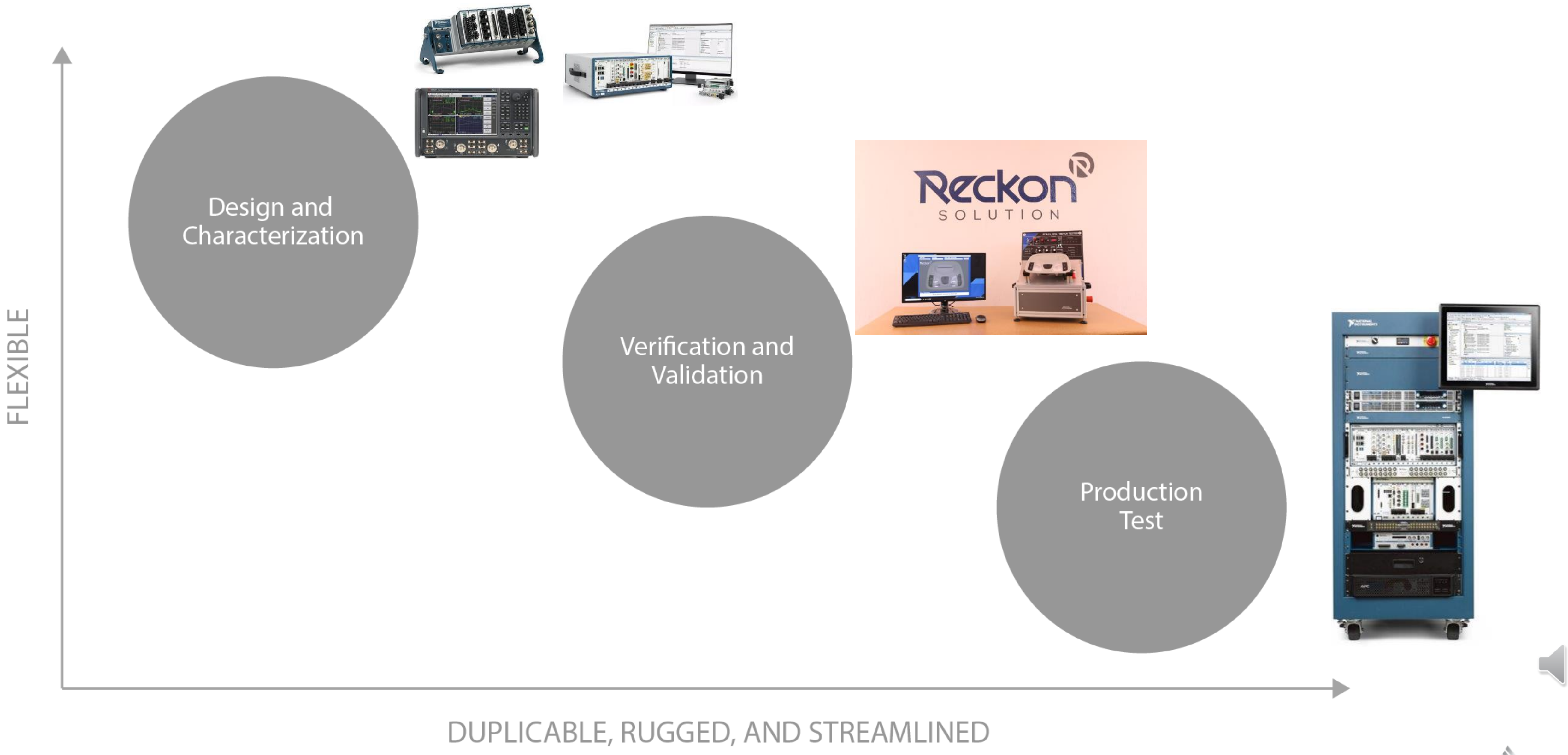
AGENDA



- Introduction
- Importance of Test System Optimization
- Consideration & Best Practices
  - Project Management
  - Hardware
  - Software
  - Case Studies
    - Global validation team
    - Reckon Lidar ATE
- Conclusion
- Q&A

Why Test Systems Are Extremely Important?

Test System Depends On Product Phase



FLEXIBLE

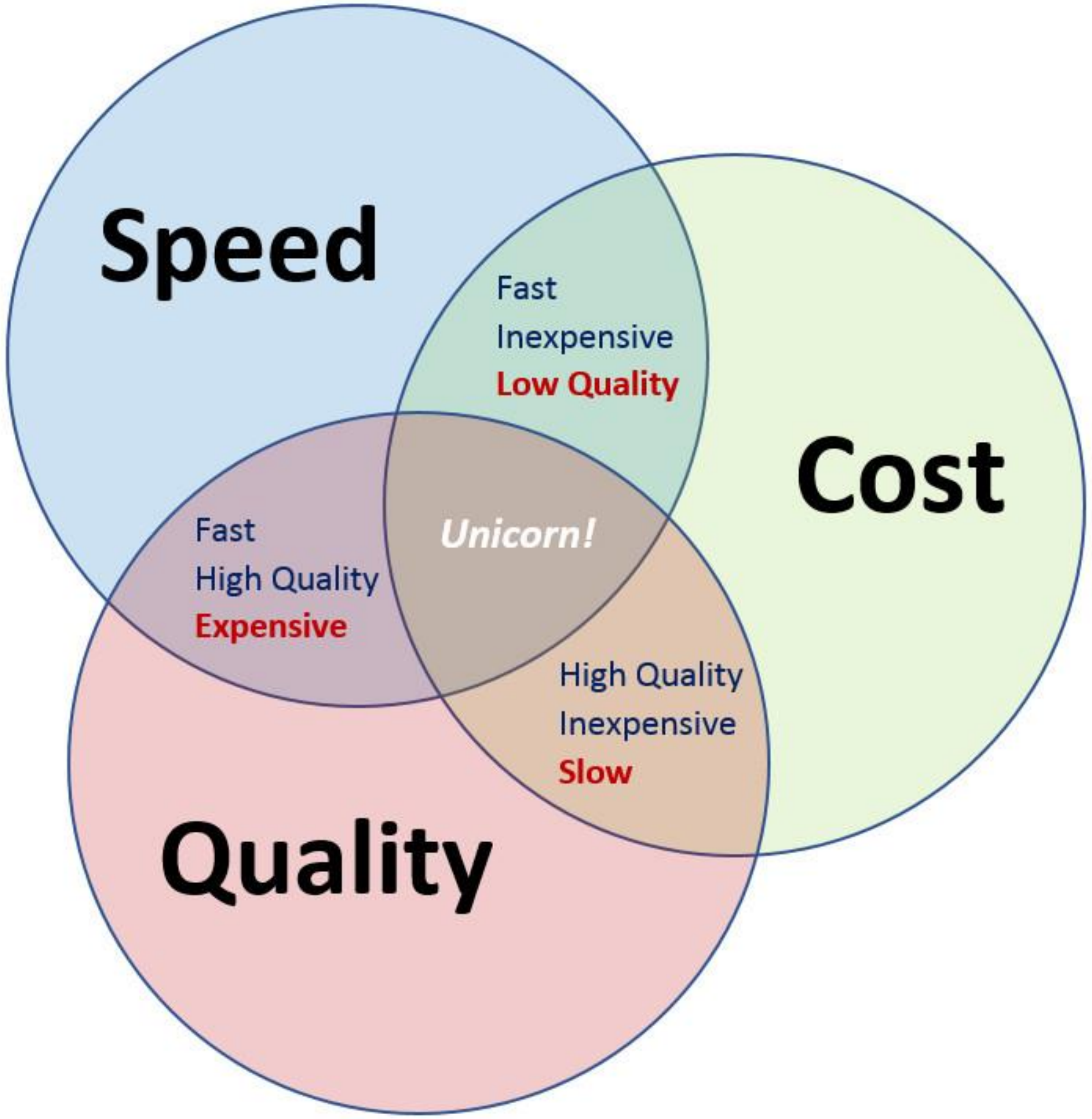
Design and Characterization

Verification and Validation

Production Test

DUPLICABLE, RUGGED, AND STREAMLINED

Main Factors At The Time To Plan A Test System



Speed

Cost

Quality

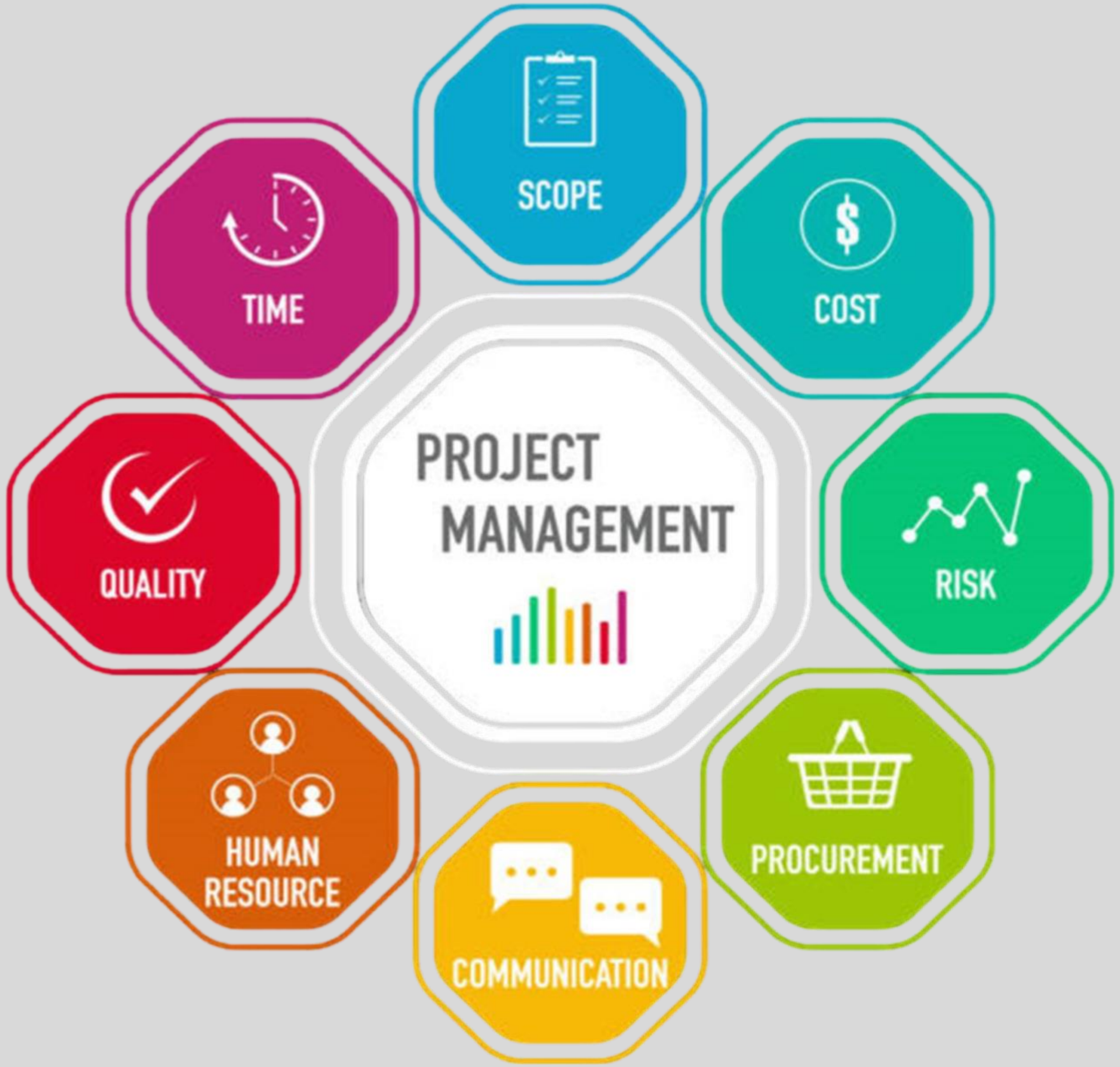
Fast
Inexpensive
Low Quality

Fast
High Quality
Expensive

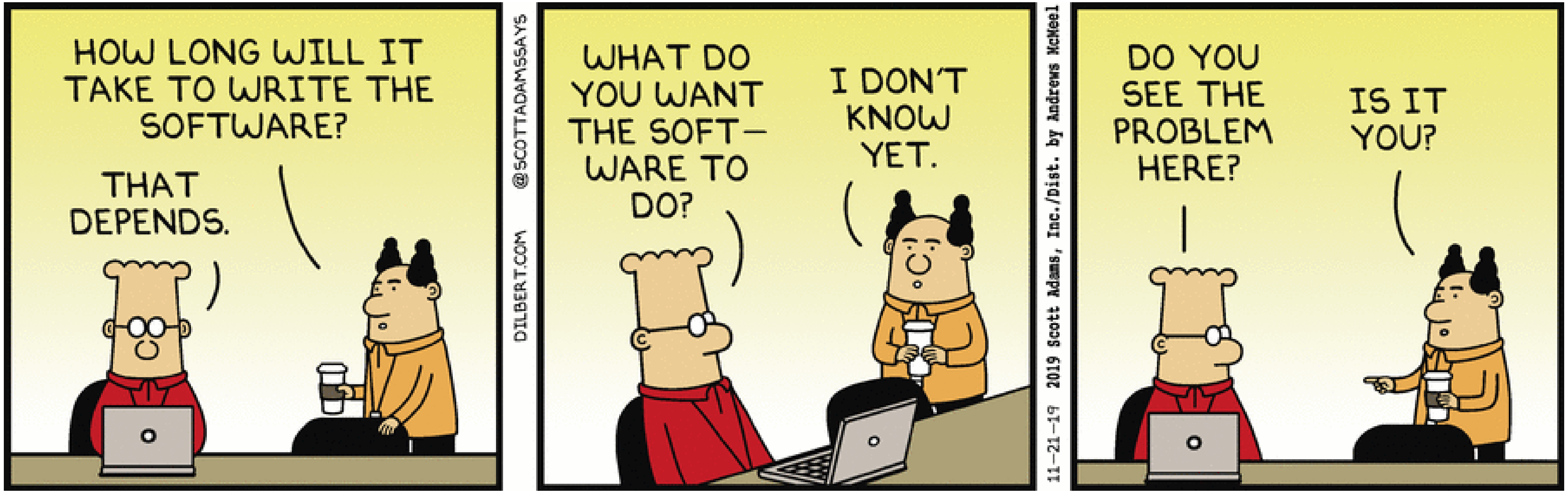
High Quality
Inexpensive
Slow

Unicorn!

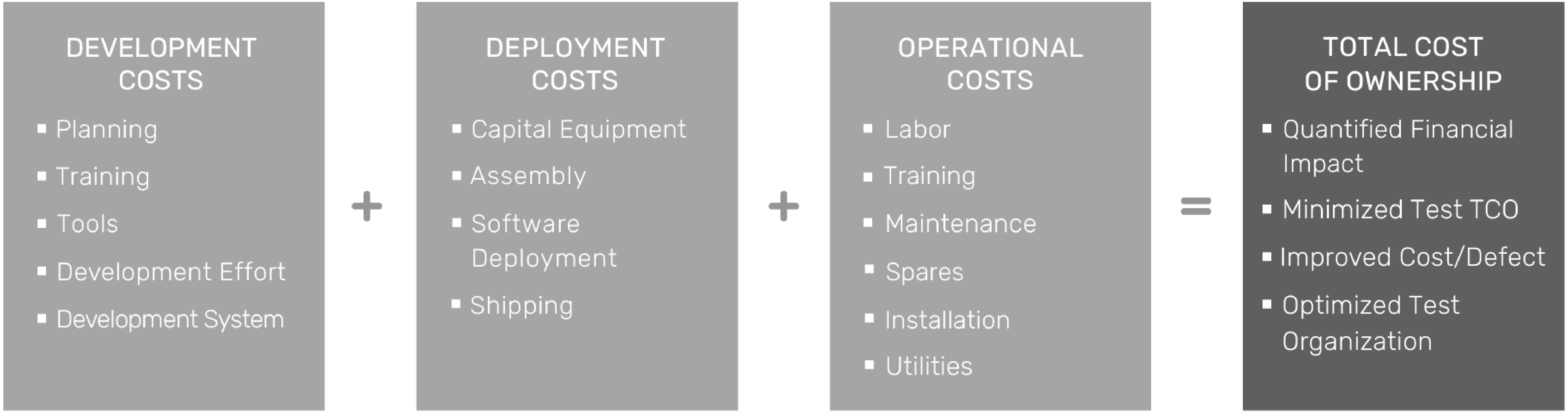
Key Areas When Develop A Test System



Collecting The Test Requirements And Defining The Project Scope Is Really Important!!!.


HOW LONG WILL IT TAKE TO WRITE THE SOFTWARE?
THAT DEPENDS.
@SCOTTADAMSSAYS
DILBERT.COM
WHAT DO YOU WANT THE SOFT-WARE TO DO?
I DON'T KNOW YET.
11-21-19 ©2019 Scott Adams, Inc./Dist. by Andrews McMeel
DO YOU SEE THE PROBLEM HERE?
IS IT YOU?

Total Cost of Ownership(TCO) an ATE (Automated Test System).



DEVELOPMENT COSTS

- Planning
- Training
- Tools
- Development Effort
- Development System

+

DEPLOYMENT COSTS

- Capital Equipment
- Assembly
- Software Deployment
- Shipping

+

OPERATIONAL COSTS

- Labor
- Training
- Maintenance
- Spares
- Installation
- Utilities

=

TOTAL COST OF OWNERSHIP

- Quantified Financial Impact
- Minimized Test TCO
- Improved Cost/Defect
- Optimized Test Organization

Hardware Considerations & Best Practices

Selecting Instrumentation

“pick the right tool for the job.”

Analog Instruments Categories

	DC AND POWER	LOW-SPEED ANALOG	HIGH-SPEED ANALOG	RF AND WIRELESS
Input, Measure	Digital Multimeter	Analog Input, Data Acquisition (DAQ)	Oscilloscope, Frequency Counter	RF Analyzer Power Meter (Spectrum Analyzer, Vector Signal, Analyzer)
Output, Generate	Programmable Power Supply	Analog Output	Function/Arbitrary Waveform Generator (FGEN, AWG)	RF Signal Generator (Vector Signal Generator, CW Source)
Input and Output On The Same Device	DC Power Analyzer	Multifunction Data Acquisition (DAQ)	All-in-One Oscilloscope	Vector Signal Transceiver (VST)
Input and Output On The Same Pin	Source Measure Unit (SMU)	LCR Meter	Impedance Analyzer	Vector Network Analyzer (VNA)

- Highly recommended to select devices with good longevity in the market, this minimize the risk to have problems for maintenance and get spare parts.

Selecting Instrumentation

“pick the right tool for the job.”

Digital Instruments Categories

<table>
<tr><th></th><th>STATIC, LOW SPEED</th><th>SYNCHRONOUS AND HIGH-SPEED PARALLEL (100 MBITS/S RANGE)</th><th>HIGH-SPEED SERIAL (10 GBITS/S RANGE)</th></tr>
<tr><td>Interface (Standard)</td><td colspan="2">Low-Speed Standard Interface Card (I2C, C) Synchronous Protocol Interface (ARINC 429, CAN, GPIB, I2C, SPI)</td><td>Interface Card (10 Gigabit Ethernet, Fiber Channel, PCI Express, and so on)</td></tr>
<tr><td>Interface (Custom)</td><td>Digital I/O (GPIO)</td><td>Digital Waveform Generator/ Analyzer, Pattern Generator</td><td>FPGA-Based High-Speed Serial Interface Aurora, Serial Rapid I/O, JESD204b/c</td></tr>
<tr><td>Electrical Test And Timing Test (Basic Interface)</td><td colspan="2">Pin Electronics Digital, Per-Pin Parametric Measurement Unit (PPMU)</td><td>BERT, Oscilloscope</td></tr>
</table>

- Highly recommended to select devices with good longevity in the market, this minimize the risk to have problems for maintenance and get spare parts.

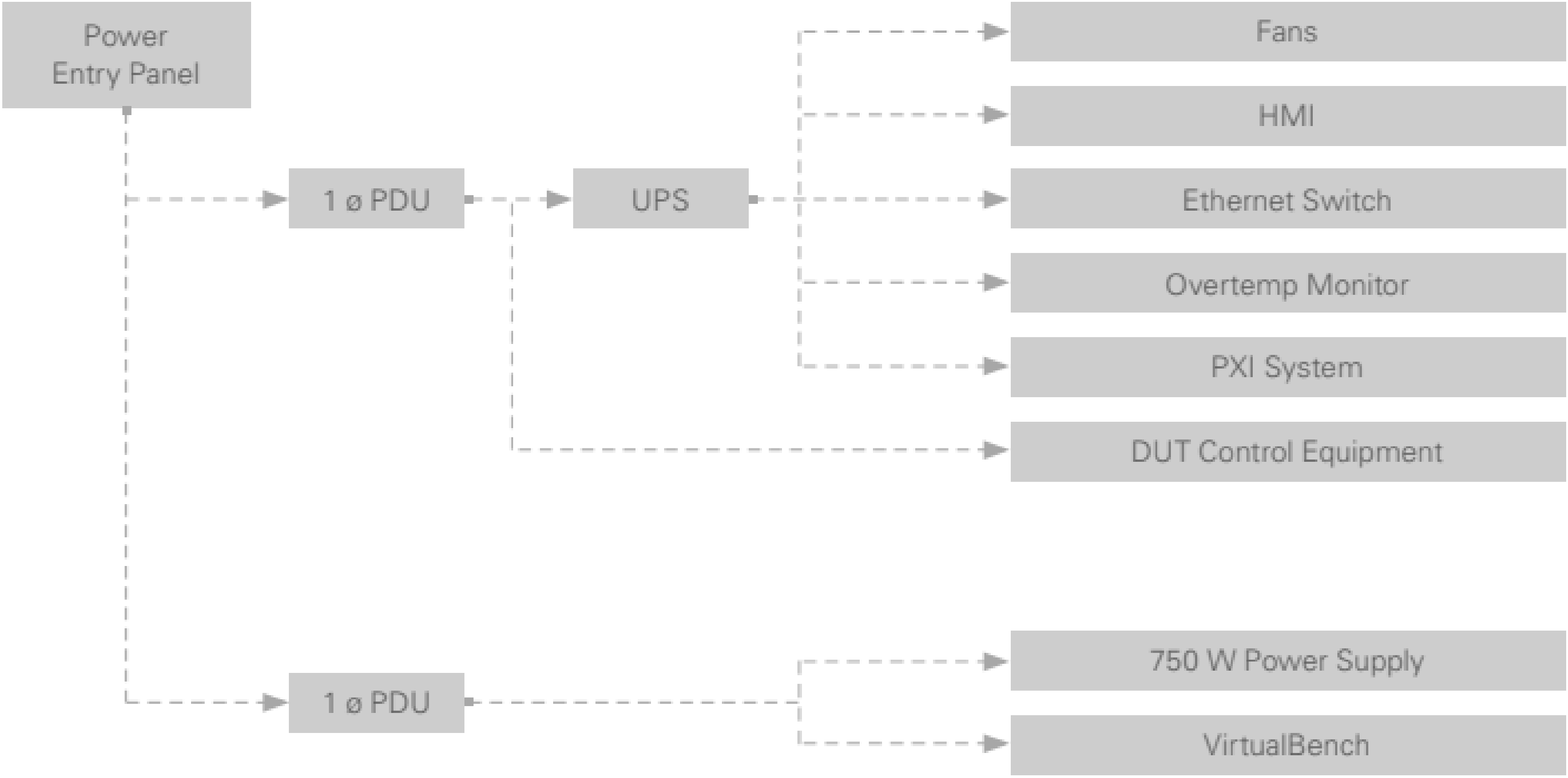
Selecting Instrumentation

“pick the right tool for the job.”

The Size Matters!

	BANDWIDTH (MBYTES/S)	LATENCY (µS)	RANGE (M) (WITHOUT EXTENDERS)	SETUP AND INSTALLATION	CONNECTOR RUGGEDNESS
GPIB	1.8 (488.1) 8 (HS488)	30	20	Good	Best
USB	60 (USB 2.0) 640 (USB 3.0)	Analog Output	5	Best	Good
PCI (PXI)	132	0.7	Internal PC Bus	Better	Better Best (for PXI)
PCI Express & PXI Express	250 (x1) 4,000 (x16)	0.7 (x1) 0.7 (x4)	Internal PC Bus	Better	Better Best (for PXI)
Ethernet/LAN/LXI	12.5 (Fast) 125 (Gigabit)	1,000 (Fast) 1,000 (Gigabit)	100 m	Good	Good

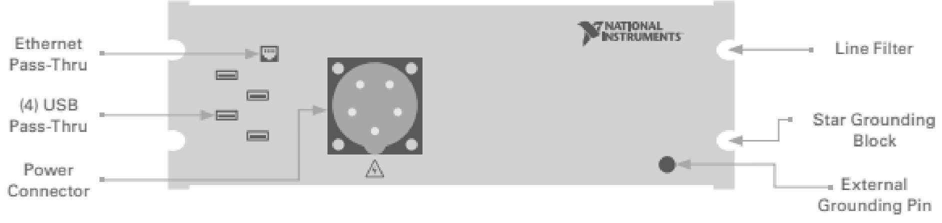
Automated Test System Power Infrastructure

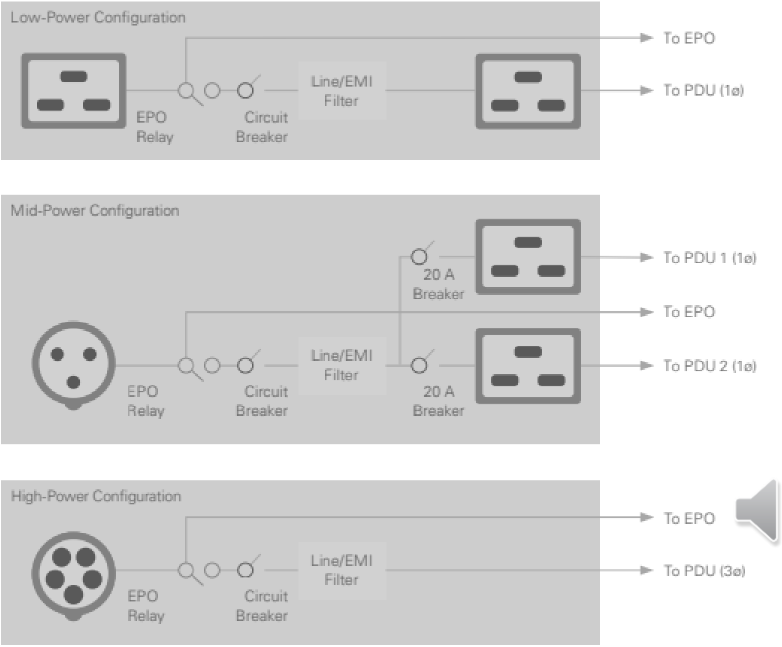


Low-Power Configuration
- IEC 60320 (C20)
- 1ø (16 A) 100-240 V

Medium-Power Configuration
- IEC 60309
- 1ø (32 A) 100-240 V

High-Power Configuration
- IEC 60309
- 3ø (16 A) 380-480 V (Red:3P+N+E)





Automated Test System Power Infrastructure

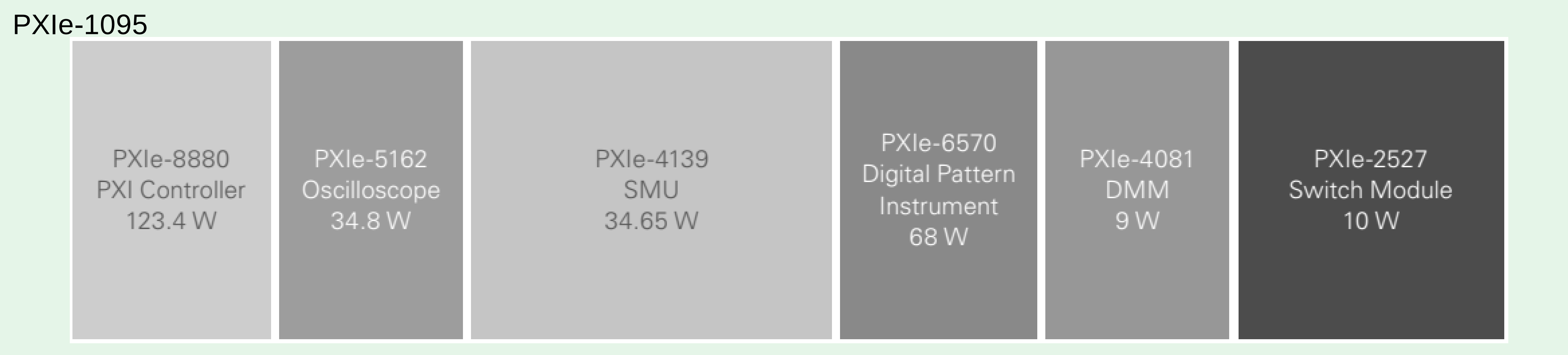
Power Budget

Equipment		Maximum Power Consumption	Average Power Utilization	Current at 110 V
PDU 1	Fans	50 W	35 W	0.03 A
	HMI	100 W	70 W	0.06 A
	Ethernet Switch	25 W	17.5 W	0.02 A
	Overtemp Monitor	10 W	7 W	0.01 A
	PXI System	526.9 W	369 W	3.4 A
	DUT Control Pumps	1,000 W	700 W	6.4 A
PDU 1 Total		-	**1,198.5 W**	**11.0 A**
PDU 2	VirtualBench	150 W	105 W	1.0 A
	750 W Power Supply	1,100 W	770 W	7.0 A
PDU 2 Total		-	**875 W**	**8.0 A**
System Total		-	**2,073.5 W**	**19.0 A**

1. Base your system power requirements on about 60 to 70 percent of the maximum required power of each component
2. Add about 20 percent to the final power calculation from rule one as a safety buffer to account for high-activity periods and any necessary future expansion of the test system.
3. Remember that some items connect through PDUs and UPSs, so there are power subsystems within the larger system.

Automated Test System Power Infrastructure

Power Budget



From→ PXIe-1095 Electrical Specifications

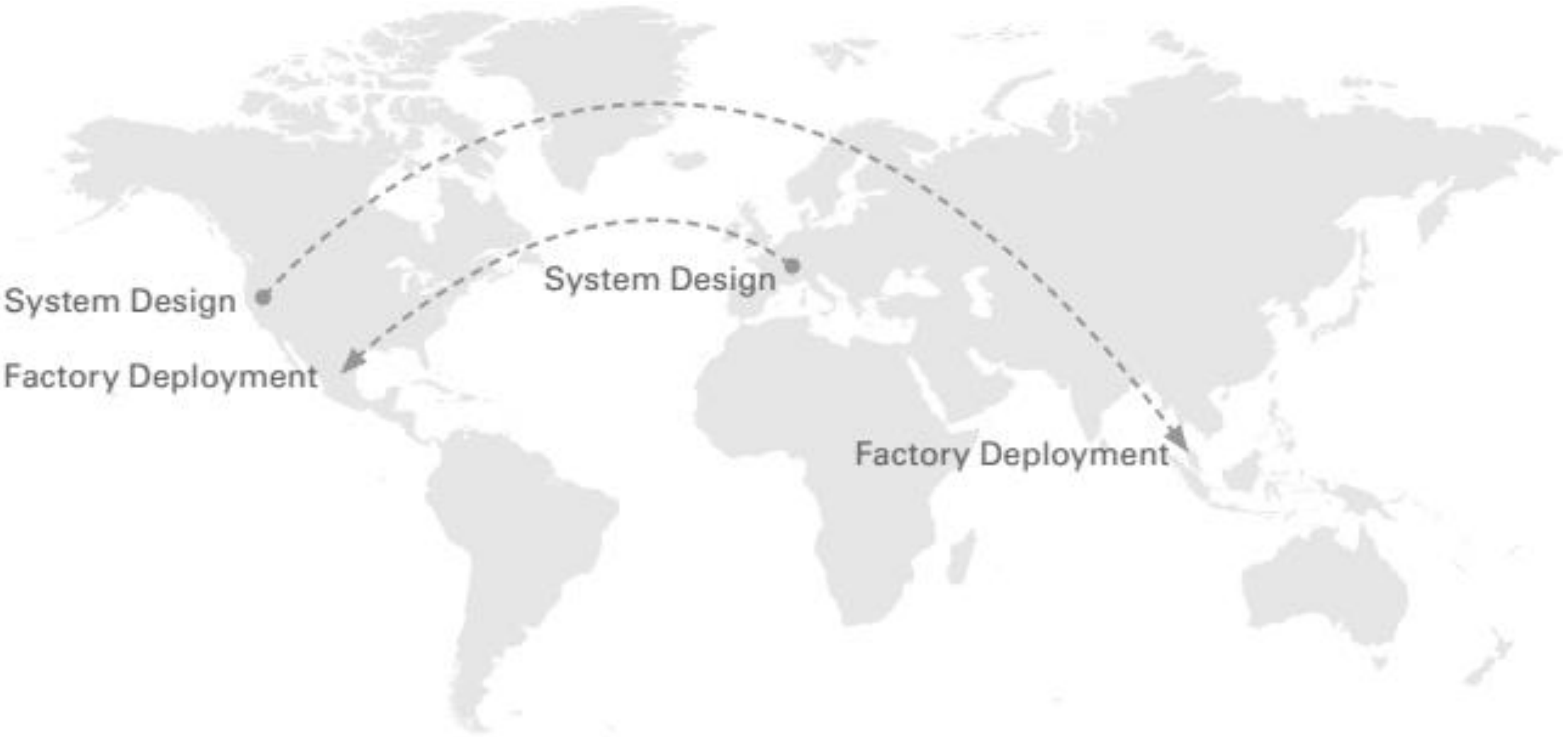
Voltage Rail	Maximum Current, Single Power Supply	Maximum Current, Dual Power Supplies
+5V_AUX	21 W	21 w
+12 V	900 W	1464 W
+5 V	107.5 W	107.5 W
+3.3 V	198 W	198 W
-12 V	15.6 W	15.6 W

From→ PXIe-8880 Electrical Specifications

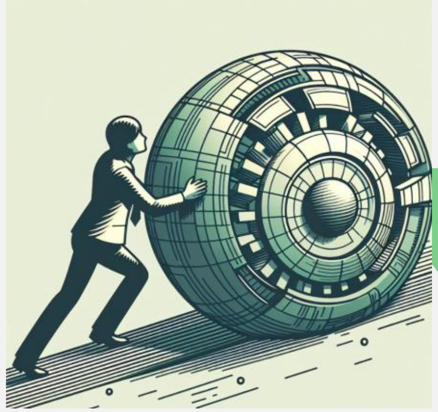
Voltage Rail (V)	Current (Amps) Typical	Current (Amps) Maximum
5 V_{AUX}	0.95 W	1.15 W
+12 V	78 W	104.4 W
+5 V	12.2 W	15.55 W
+3.3 V	7.4 W	9.8 W

Automated Test System Power Infrastructure

Check List	Item
☑	Power grid voltage standard and configuration
☑	Power grid quality and reliability
☑	Materials compliance like RoHS
☑	Energy compliance like CE, PSE, or KC
☑	Trade compliance and import/export regulations



Automated Test System Power Infrastructure Best Practices For Components



Working closely with a vendor would allow you to identify direct replacements for EOL devices, as well as new products on the pipeline.

Consider younger devices on the market where the EOL is not nearby.

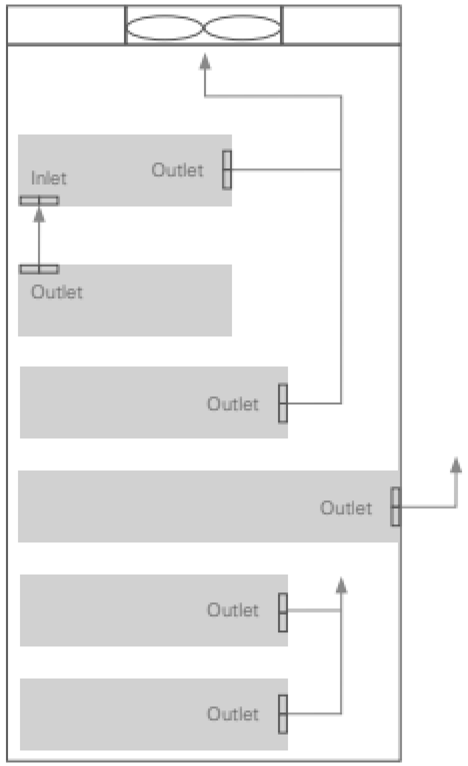
It is a best practice to consider standard components rather than creating custom parts.

Sourcing commercially available components from an established vendor is a longer-term strategy rather than creating custom parts.

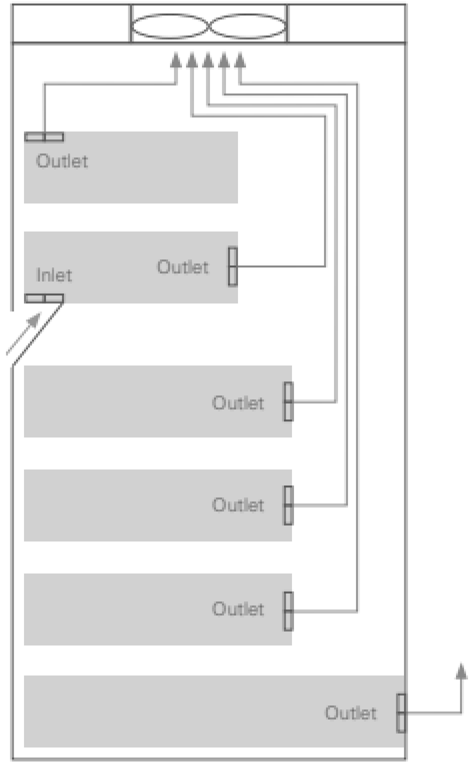
Design and plan your system for future expansion.

Rack Layout and Thermal Profiling

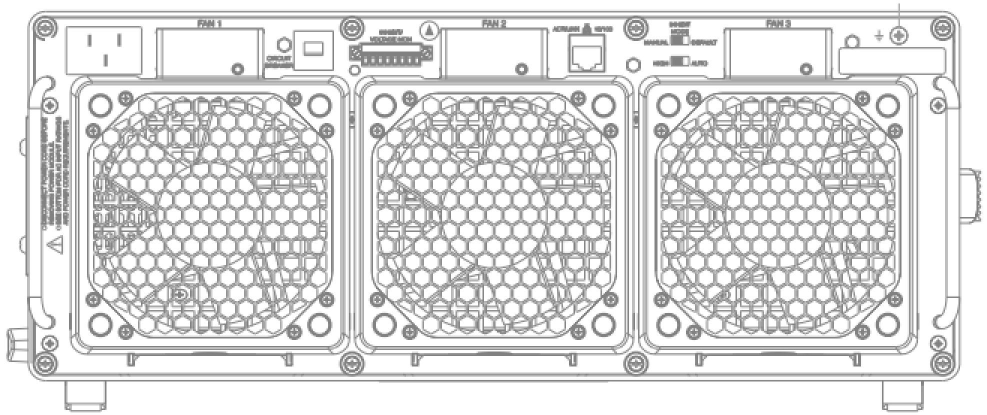
System Layout Instrument Blocking Airflow



System Layout Instrument Proper Airflow

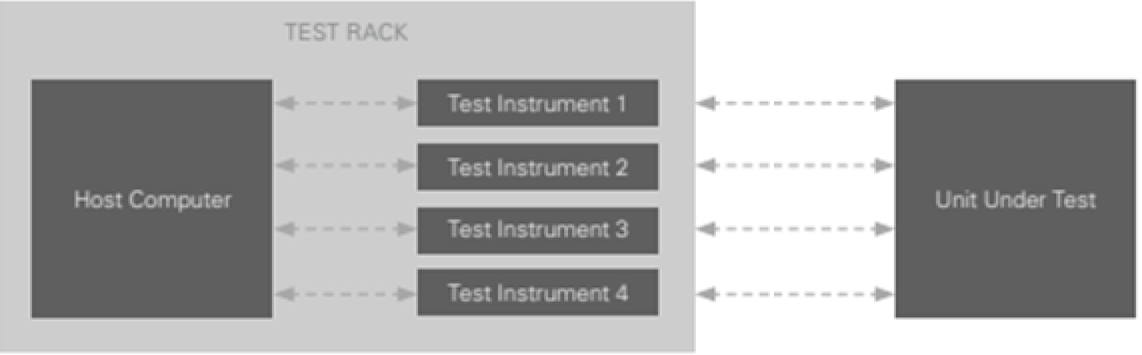


PXIe Chassis Cooling System

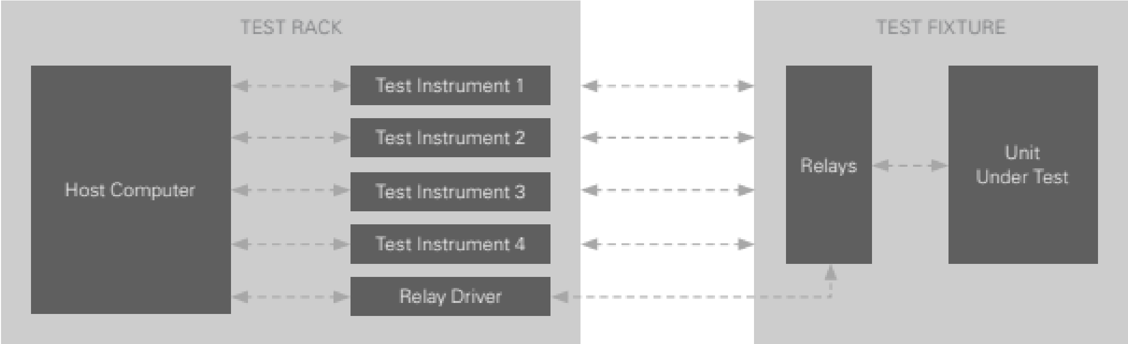


Switching and Multiplexing

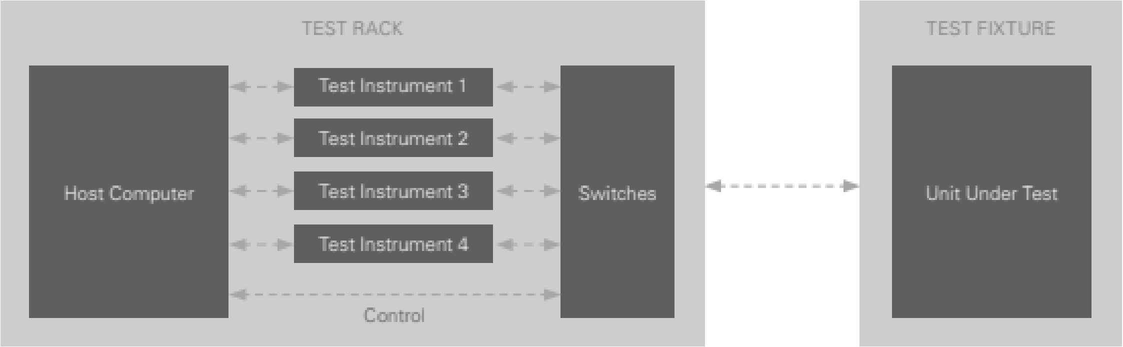
No Switching



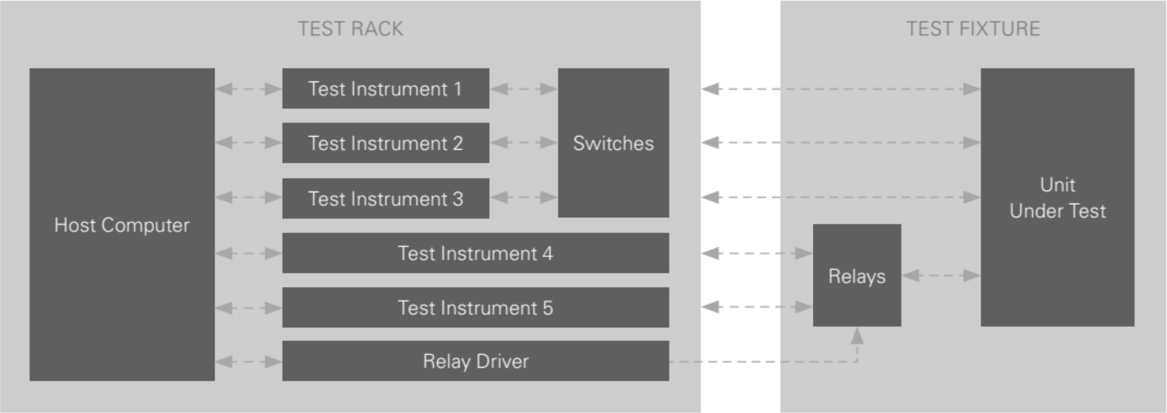
Switching In Fixture Only



Switching In Test Rack Only



Switching In Test Rack & Fixture



Switching and Multiplexing

Below ○ Average ◒ Above ●

	FLEXIBILITY	THROUGHPUT	COST	LOW-LEVEL MEASUREMENTS (MV, µA, MΩ)
No Switching	○	●	○	●
Switching in Test Rack	●	◒	●	○
Switching in Test Fixture	○	◒	◒	◒
Switching in Test Rack and Fixture	●	◒	◒	◒

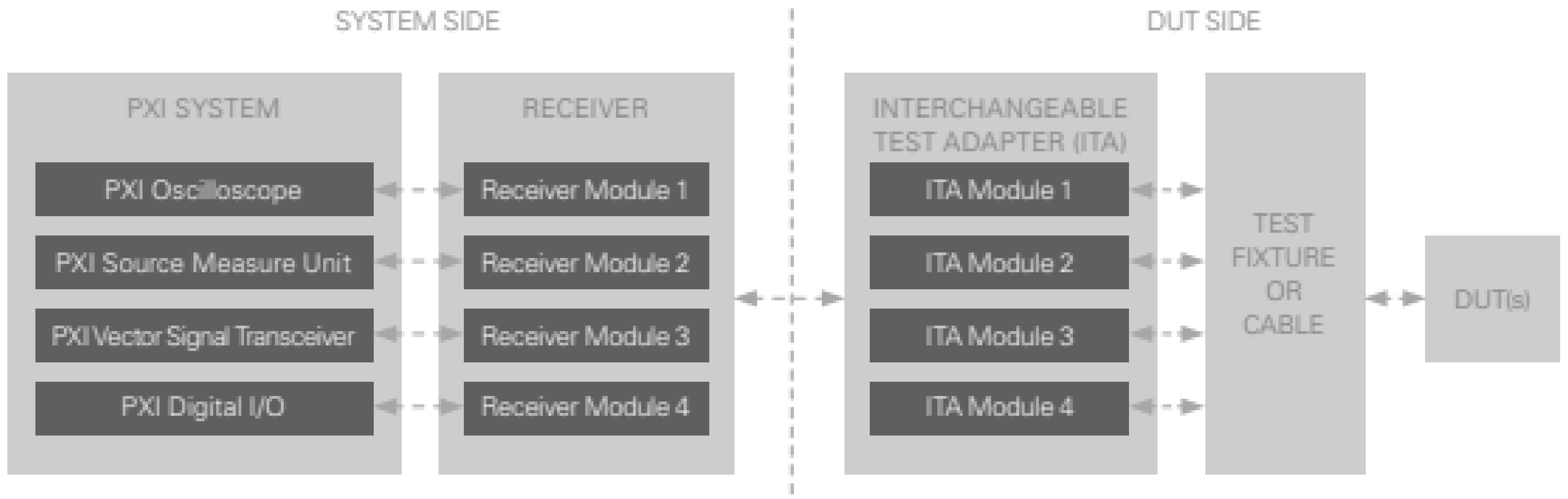
Switching and Multiplexing

Selecting The Right Relay Option For Our Test Solution Is Crucial”

Below ○ Average ◒ Above ●

CAPABILITY	Electromechanical Relay (EMR)	Reed Relay	Field-Effect Transistor (FET)	Solid State Relay(SSR)
High Power	●	◒	○	◒
High Speed	○	◒	●	◒
Small Package Size	◒	●	●	●
Low Path Resistance	●	◒	○	◒
Low Voltage Offset	◒	○	◒	●
Extended Lifetime	○	◒	●	●

Mass Interconnect and Fixturing



SYSTEM SIDE

DUT SIDE

PXI SYSTEM

- PXI Oscilloscope
- PXI Source Measure Unit
- PXI Vector Signal Transceiver
- PXI Digital I/O

RECEIVER

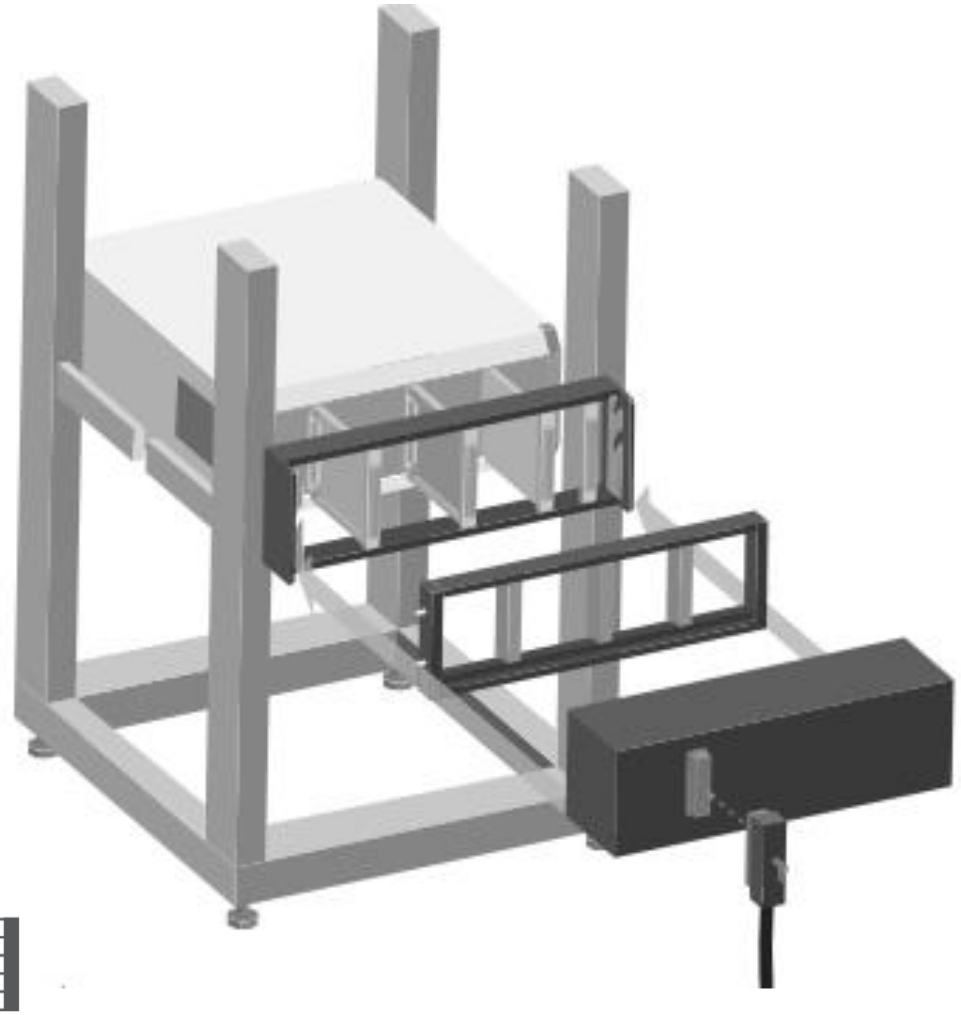
- Receiver Module 1
- Receiver Module 2
- Receiver Module 3
- Receiver Module 4

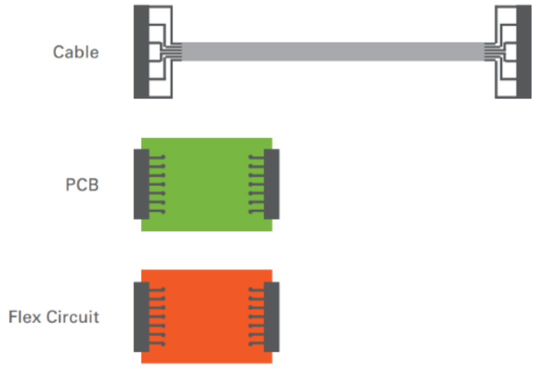
INTERCHANGEABLE TEST ADAPTER (ITA)

- ITA Module 1
- ITA Module 2
- ITA Module 3
- ITA Module 4

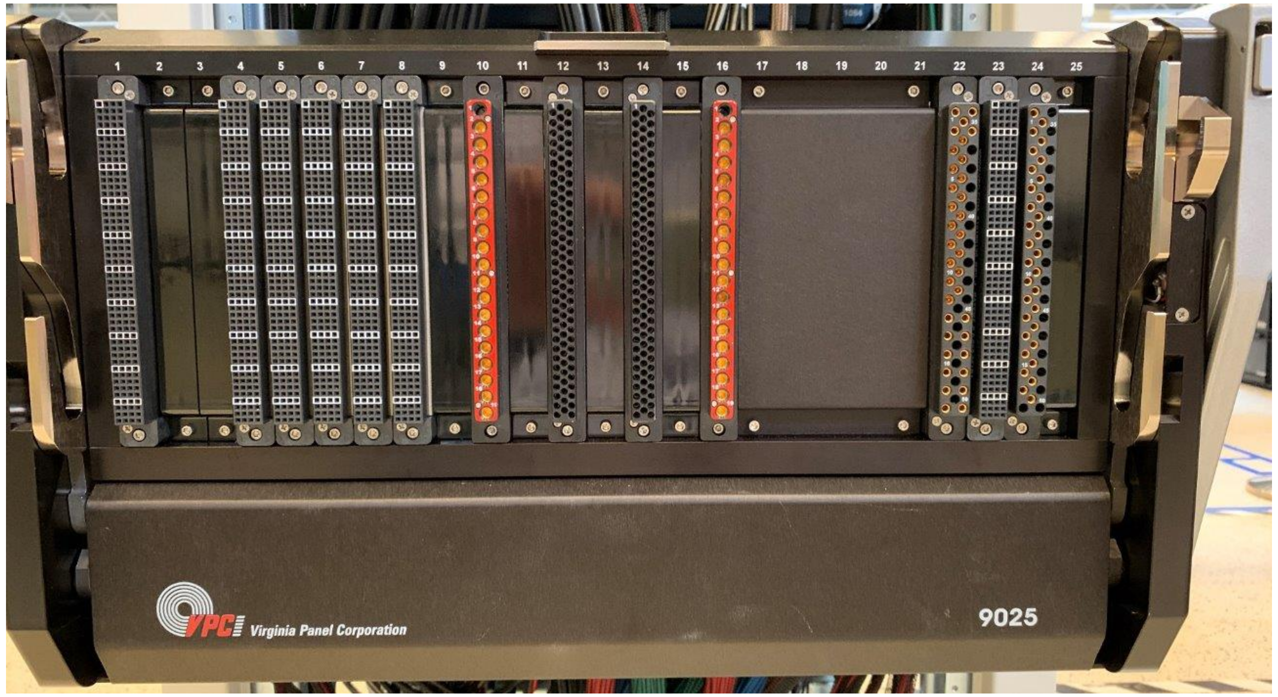
TEST FIXTURE OR CABLE

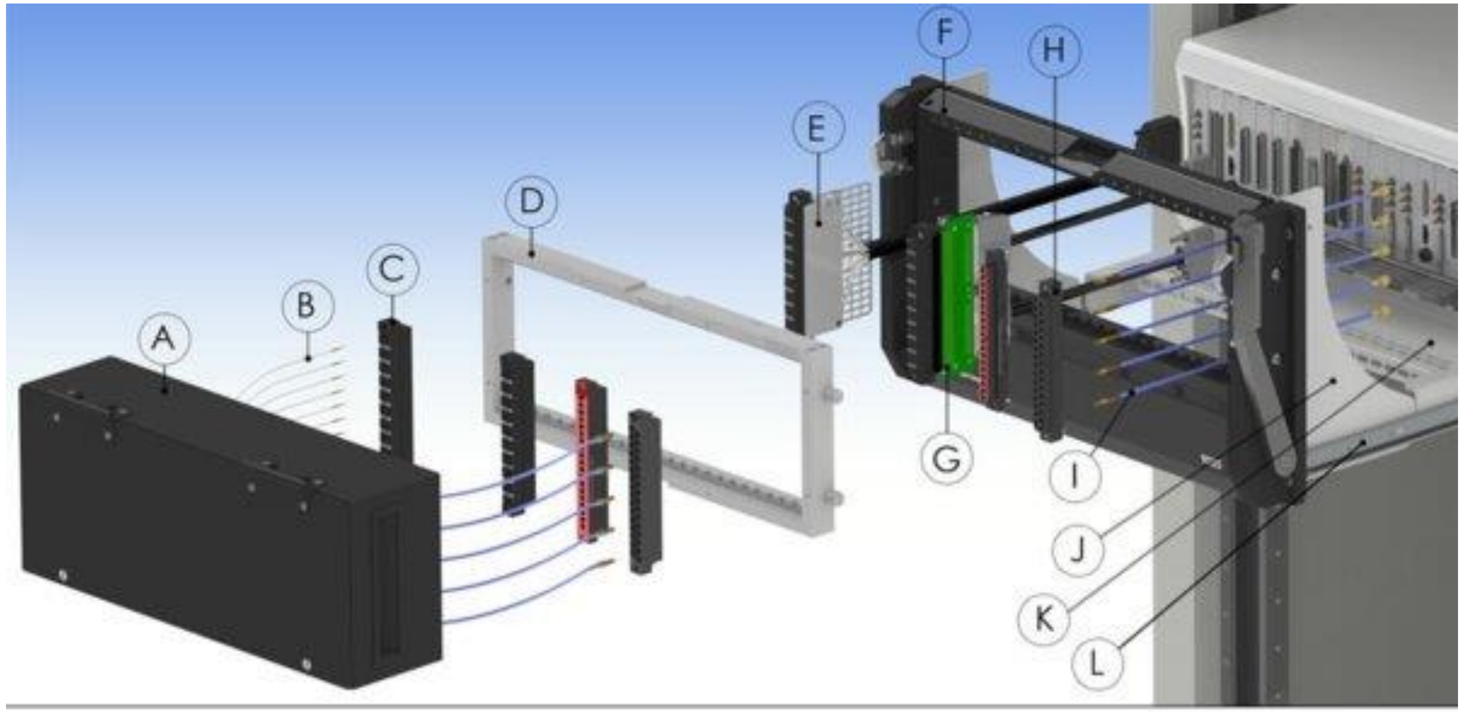
DUT(s)





Mass Interconnect and Fixturing





A. ITA Enclosure
B. ITA Patchcords
C. ITA Modules
D. 9025 ITA

E. Receiver Cable Assembly
F. 9025 Receiver
G. Receiver PCB Adapter
H. Receiver Module

I. Receiver Patchcords
J. Mounting Flanges
K. Instrument Bracket
L. Slide Kit

Hardware Considerations To Build a Fixture

Below ○ Average ◒ Above ●

ABSTRACTION OPTION	CABLES	MASS INTERCONNECT WITH CABLES	MASS INTERCONNECT WITH PCBS OR FLEX CIRCUITS
Frequent Changeover Between DUTs	○	●	●
Optimized For Design And Characterization	●	○	○
Optimized For Verification And Validation (V&V)	◒	◒	◒
Optimized For Test Production	○	●	●
Signal Quality	◒	◒	●
Continuity of Performance (system to system)	◒	◒	●
Ease of System Maintenance And Upgradability	○	●	●
System Reconfiguration (that is, scalability)	○	●	●
Ease of Duplication (for example, global deployments)	○	◒	●
Instrument To Module Pin Efficiency	○	●	◒
Repairability In The Field	●	●	○
Instrument Card Rev. Control Tolerance	●	●	○

System Maintenance

DESIGN GUIDELINES	PREDICTIVE	PREVENTIVE	CORRECTIVE
Self-Test and Monitoring	• Condition monitoring • Verifying functionality	• Verifying functionality	• Detecting failures • Diagnosing and localizing failures • Verifying functionality
Modular Design	• Condition monitoring • Servicing • Replacing • Calibrating • Verifying functionality	• Servicing • Replacing • Calibrating • Verifying functionality	• Detecting failures • Diagnosing and localizing failures • Repairing • Verifying functionality
Standardization	• Condition monitoring • Servicing • Replacing • Calibrating • Verifying functionality Improving consistency of work	• Servicing • Replacing • Calibrating • Verifying functionality • Improving consistency of work	• Detecting failures • Diagnosing and localizing failures • Repairing • Verifying functionality • Improving consistency of work
Simplicity	• Lowering documentation and training costs • Improving consistency of work	• Lowering documentation and training costs • Improving consistency of work	• Lowering documentation and training costs • Improving consistency of work
Environment and Human Factors	• Lowering frequency of predictive maintenance events • Reducing human errors Improving safety	• Lowering frequency of preventive maintenance events • Reducing human errors • Improving safety	• Lowering failure rates and • Reducing human errors • Improving safety

Software Considerations & Case Studies

Case Study 1: Global validation lab

- Multiple validation labs within a global semiconductor company are using their workbenches to automate their manual validation.
- This company recently switched from a DIY solution that each validation lab implemented, normally by a single person that took the initiative, to the use of TestStand as a company-enforced test executive.
- Now, all validation labs have a centralized location where all sequences can be accessed, which are maintained by a team of 200+ validation engineers spread across multiple facilities.
- This new approach of using TestStand as a standard for automation has greatly increased profitability due to the number of devices under test that can be passed through the test sequence, allowing management to schedule more projects due to the increased bandwidth.
- As time goes by… product increases in complexity. Management is now having to push back again on validating new silicon due to issues on replicating validation results across multiple labs, as well as not being able to validate parts fast enough.

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *“If it works… don’t touch it!”.*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a “file share” only, there is no way of knowing if a remote has a working copy.

5.- Not making use of commercially available deployment tools

It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Coupling and technical debt

- An automation expert has been brought into the company. Upper management then gave him full control on defining a company-wide approach for automation across all sites.

- After careful evaluation, an initiative was created that would roll out on 2 stages.

STAGE 1

- Challenging the status quo. *“This is how we have always done it”* is no longer a valid phrase.
- Identifying areas to define trainings and a mentoring program to help validation engineers ramp up on TestStand.
- Have the multiple managers from the multiple validation labs align on this plan and understand individual needs that each site has.

STAGE 2

- A mechanism to automate enforcing best practices is embedded into source code control.
- Now, if a developer does not adhere to a rule, it is automatically rejected by the global repo.
- All reuse code is then modified to adhere to these best practices, following an established workflow using source code control.

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *“If it works… don’t touch it!”.*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a “file share” only, there is no way of knowing if a remote has a working copy.

5.- Not making use of commercially available deployment tools

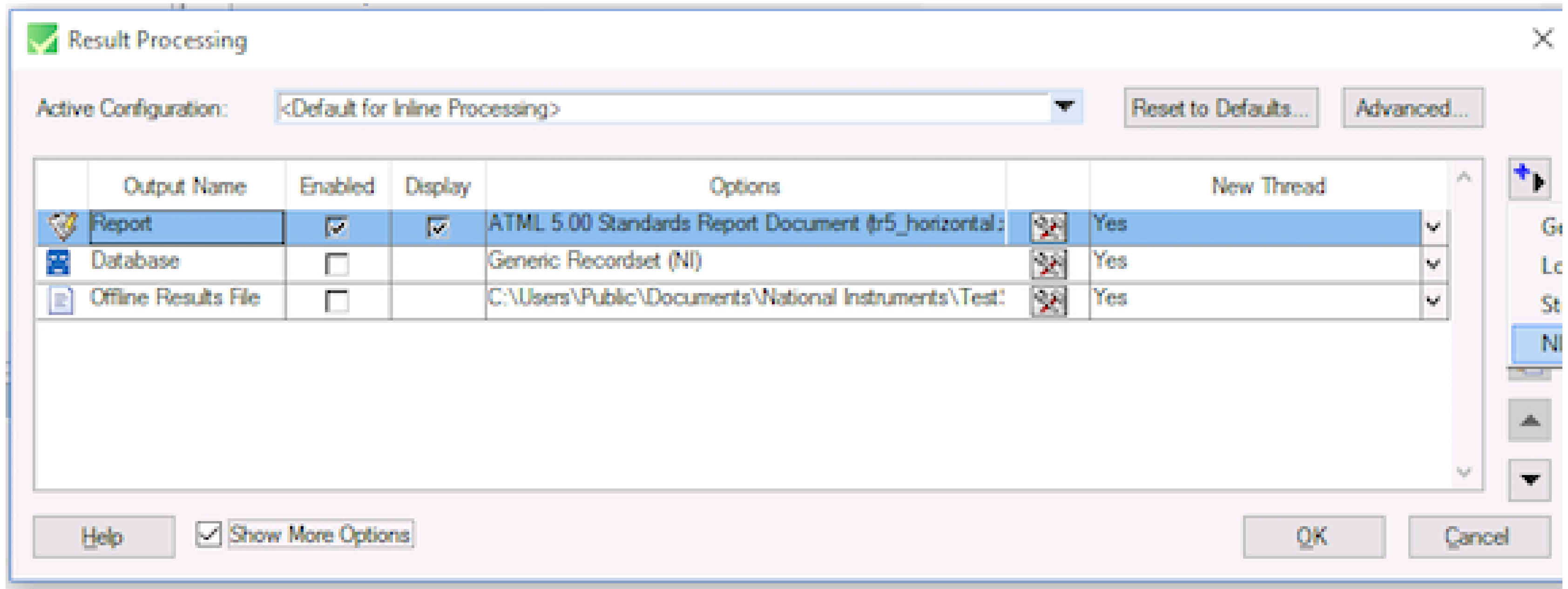
It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Result collection strategy

- **NI SystemLink enterprise** was chosen for result collection due to the company wide access to results that this product would bring.
- Enabling this result processing plugin on TestStand would automatically post results to the network, no changes to sequences are necessary if the sequence is designed to use a results processing plugin.



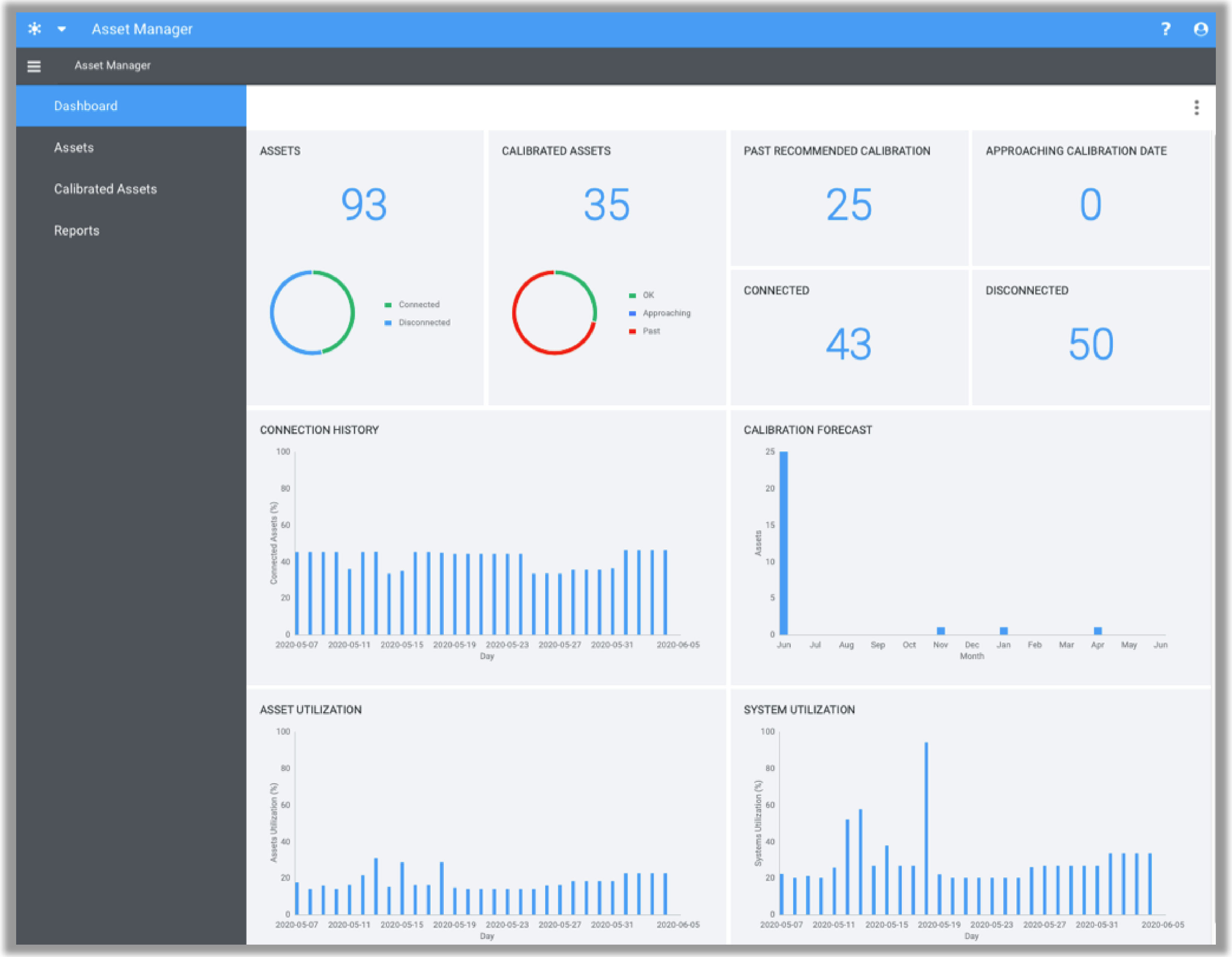
Result Processing

Active Configuration: <Default for Inline Processing>

Output Name	Enabled	Display	Options	New Thread
Report	☑	☑	ATML 5.00 Standards Report Document (tr5_horizontal.:	Yes
Database	☐		Generic Recordset (NI)	Yes
Offline Results File	☐		C:\Users\Public\Documents\National Instruments\Test!	Yes

☑ Show More Options

Device utilization and calibration forecasting



ASSETS 93

CALIBRATED ASSETS 35

PAST RECOMMENDED CALIBRATION 25

APPROACHING CALIBRATION DATE 0

CONNECTED 43

DISCONNECTED 50

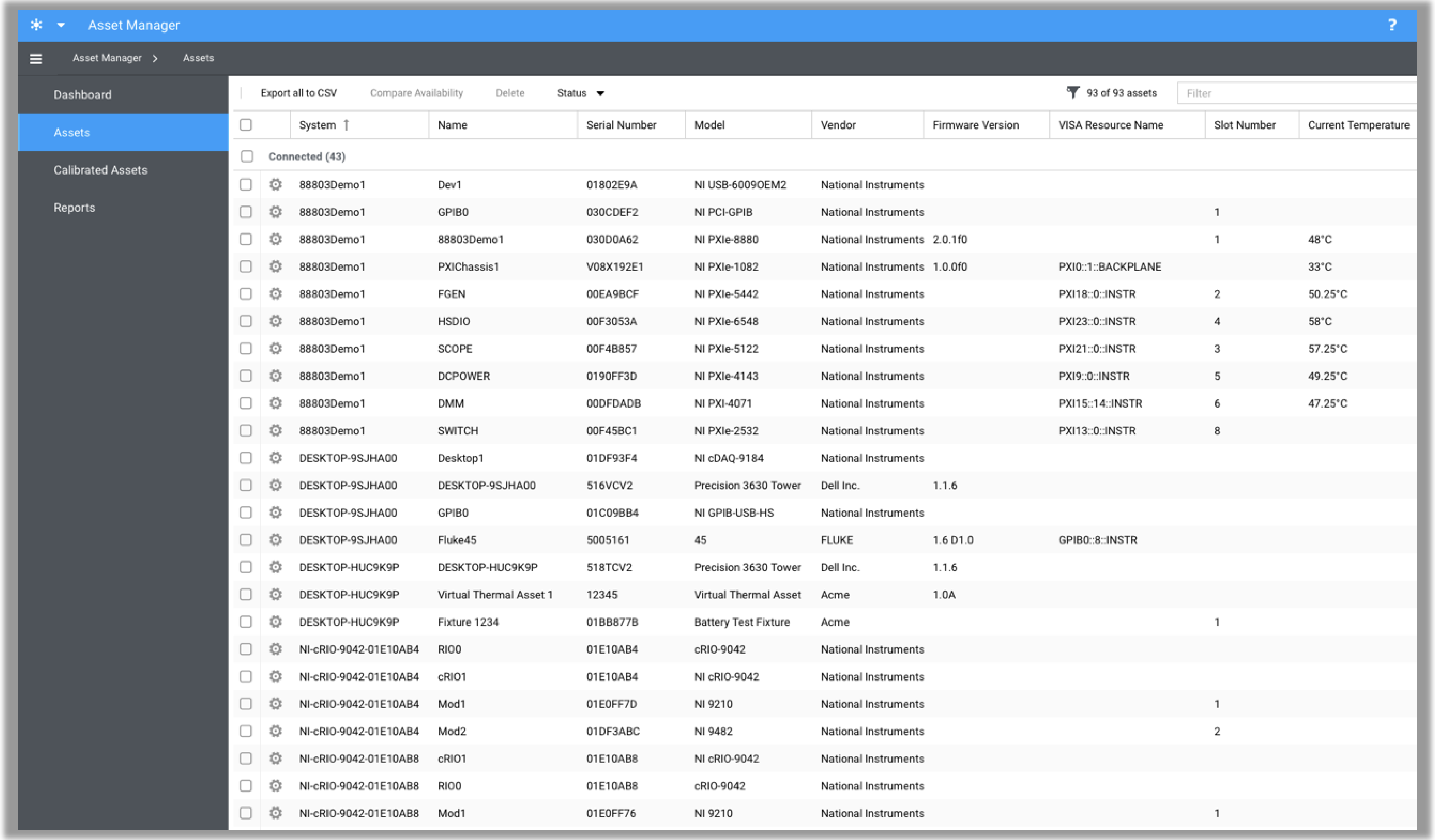
CONNECTION HISTORY

CALIBRATION FORECAST

ASSET UTILIZATION

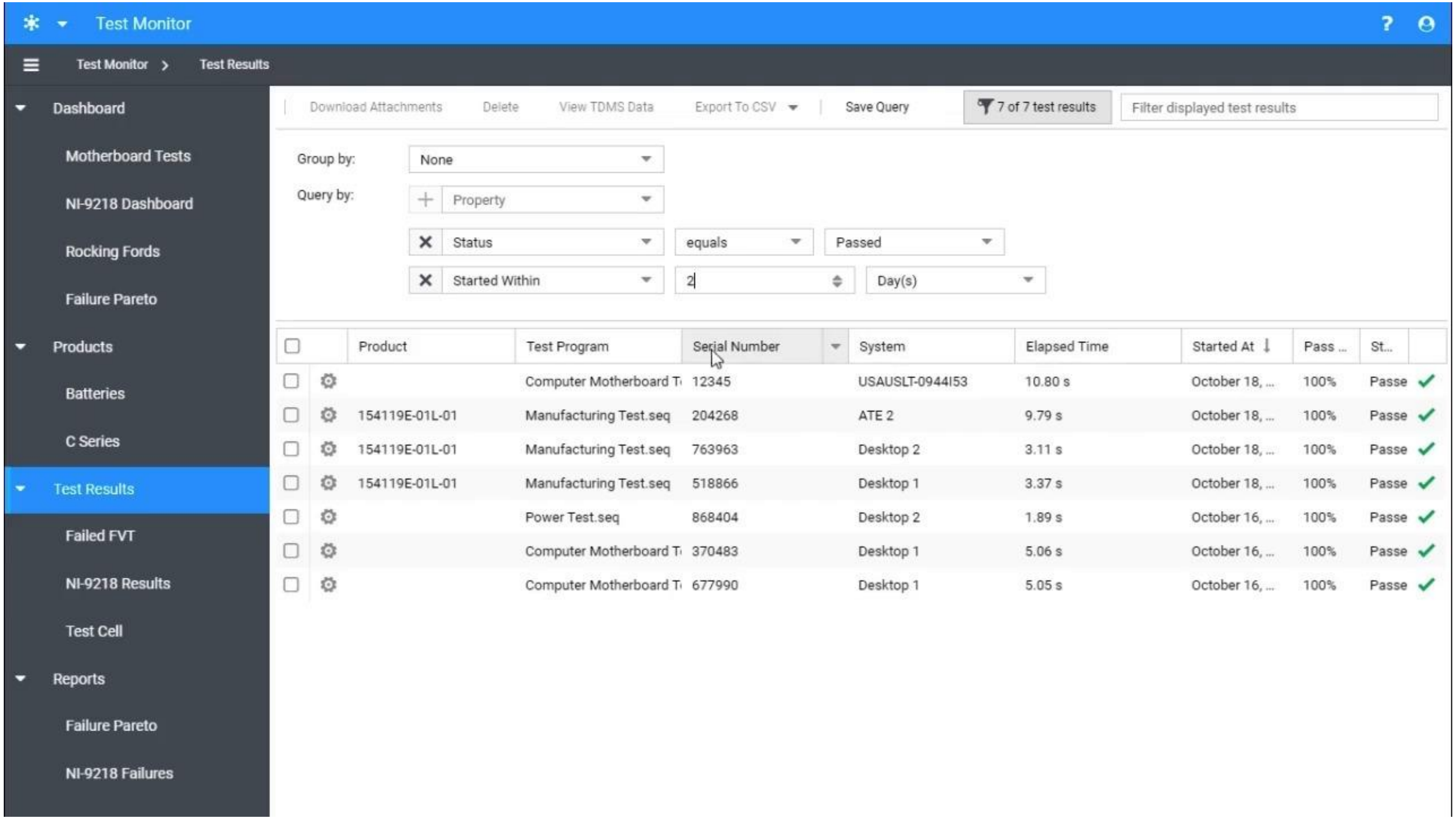
SYSTEM UTILIZATION

Asset tracking



□	System ↑	Name	Serial Number	Model	Vendor	Firmware Version	VISA Resource Name	Slot Number	Current Temperature
□	Connected (43)								
□	88803Demo1	Dev1	01802E9A	NI USB-6009OEM2	National Instruments				
□	88803Demo1	GPIB0	030CDEF2	NI PCI-GPIB	National Instruments			1	
□	88803Demo1	88803Demo1	030D0A62	NI PXIe-8880	National Instruments	2.0.1f0		1	48°C
□	88803Demo1	PXIChassis1	V08X192E1	NI PXIe-1082	National Instruments	1.0.0f0	PXI0::1::BACKPLANE		33°C
□	88803Demo1	FGEN	00EA9BCF	NI PXIe-5442	National Instruments		PXI18::0::INSTR	2	50.25°C
□	88803Demo1	HSDIO	00F3053A	NI PXIe-6548	National Instruments		PXI23::0::INSTR	4	58°C
□	88803Demo1	SCOPE	00F4B857	NI PXIe-5122	National Instruments		PXI21::0::INSTR	3	57.25°C
□	88803Demo1	DCPOWER	0190FF3D	NI PXIe-4143	National Instruments		PXI9::0::INSTR	5	49.25°C
□	88803Demo1	DMM	00DFDADB	NI PXI-4071	National Instruments		PXI15::14::INSTR	6	47.25°C
□	88803Demo1	SWITCH	00F45BC1	NI PXIe-2532	National Instruments		PXI13::0::INSTR	8	
□	DESKTOP-9SJHA00	Desktop1	01DF93F4	NI cDAQ-9184	National Instruments				
□	DESKTOP-9SJHA00	DESKTOP-9SJHA00	516VCV2	Precision 3630 Tower	Dell Inc.	1.1.6			
□	DESKTOP-9SJHA00	GPIB0	01C09BB4	NI GPIB-USB-HS	National Instruments				
□	DESKTOP-9SJHA00	Fluke45	5005161	45	FLUKE	1.6 D1.0	GPIB0::8::INSTR		
□	DESKTOP-HUC9K9P	DESKTOP-HUC9K9P	518TCV2	Precision 3630 Tower	Dell Inc.	1.1.6			
□	DESKTOP-HUC9K9P	Virtual Thermal Asset 1	12345	Virtual Thermal Asset	Acme	1.0A			
□	DESKTOP-HUC9K9P	Fixture 1234	01BB877B	Battery Test Fixture	Acme			1	
□	NI-cRIO-9042-01E10AB4	RIO0	01E10AB4	cRIO-9042	National Instruments				
□	NI-cRIO-9042-01E10AB4	cRIO1	01E10AB4	NI cRIO-9042	National Instruments				
□	NI-cRIO-9042-01E10AB4	Mod1	01E0FF7D	NI 9210	National Instruments			1	
□	NI-cRIO-9042-01E10AB4	Mod2	01DF3ABC	NI 9482	National Instruments			2	
□	NI-cRIO-9042-01E10AB8	cRIO1	01E10AB8	NI cRIO-9042	National Instruments				
□	NI-cRIO-9042-01E10AB8	RIO0	01E10AB8	cRIO-9042	National Instruments				
□	NI-cRIO-9042-01E10AB8	Mod1	01E0FF76	NI 9210	National Instruments			1	

Accessibility to results



Group by: None

Query by: Property

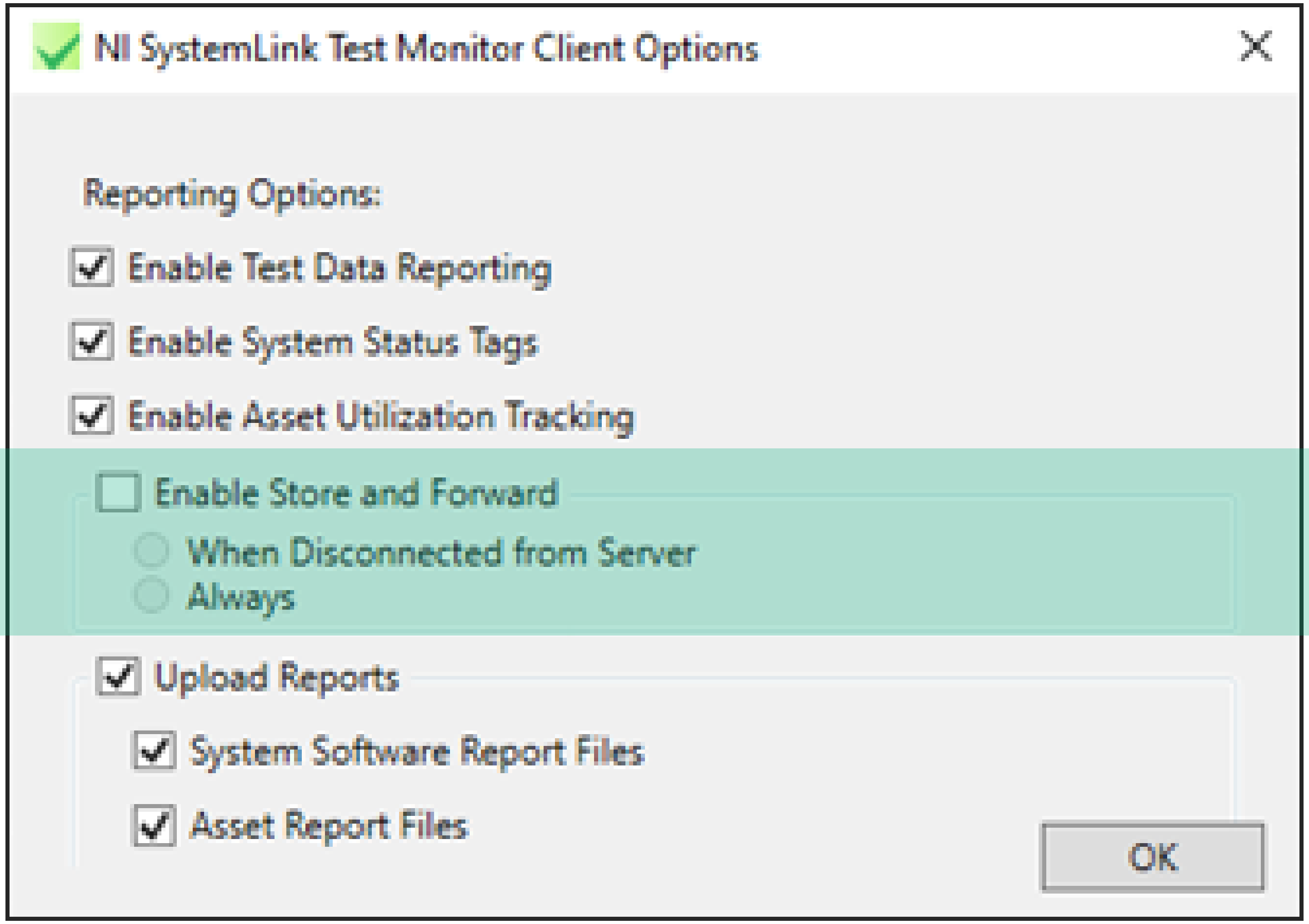
Status equals Passed

Started Within 2 Day(s)

□		Product	Test Program	Serial Number	System	Elapsed Time	Started At	Pass ...	St...	
□	⚙		Computer Motherboard Te	12345	USAUSLT-0944I53	10.80 s	October 18, ...	100%	Passe	✓
□	⚙	154119E-01L-01	Manufacturing Test.seq	204268	ATE 2	9.79 s	October 18, ...	100%	Passe	✓
□	⚙	154119E-01L-01	Manufacturing Test.seq	763963	Desktop 2	3.11 s	October 18, ...	100%	Passe	✓
□	⚙	154119E-01L-01	Manufacturing Test.seq	518866	Desktop 1	3.37 s	October 18, ...	100%	Passe	✓
□	⚙		Power Test.seq	868404	Desktop 2	1.89 s	October 16, ...	100%	Passe	✓
□	⚙		Computer Motherboard Te	370483	Desktop 1	5.06 s	October 16, ...	100%	Passe	✓
□	⚙		Computer Motherboard Te	677990	Desktop 1	5.05 s	October 16, ...	100%	Passe	✓

Network independent result collection

- “Test Monitor Client” has a service called “Store and forward”.



NI SystemLink Test Monitor Client Options

Reporting Options:

☑ Enable Test Data Reporting

☑ Enable System Status Tags

☑ Enable Asset Utilization Tracking

□ Enable Store and Forward

○ When Disconnected from Server

○ Always

☑ Upload Reports

☑ System Software Report Files

☑ Asset Report Files

No more “babysitting” parts…

- Very busy product experts were unable to delegate running multiple parts to technicians, due to the room for mistake when typing in a part number or selecting the wrong sequence.

- A modified version of the TestStand sequential process model was created that automated reading the part number via JTAG. No more manual typing.

- Additionally, a new feature was added that sent the status of the test to a mobile device. This way the very busy product expert is able to track multiple technicians running parts on multiple benches across multiple labs.

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *"If it works… don't touch it!".*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a "file share" only, there is no way of knowing if a remote has a working copy.

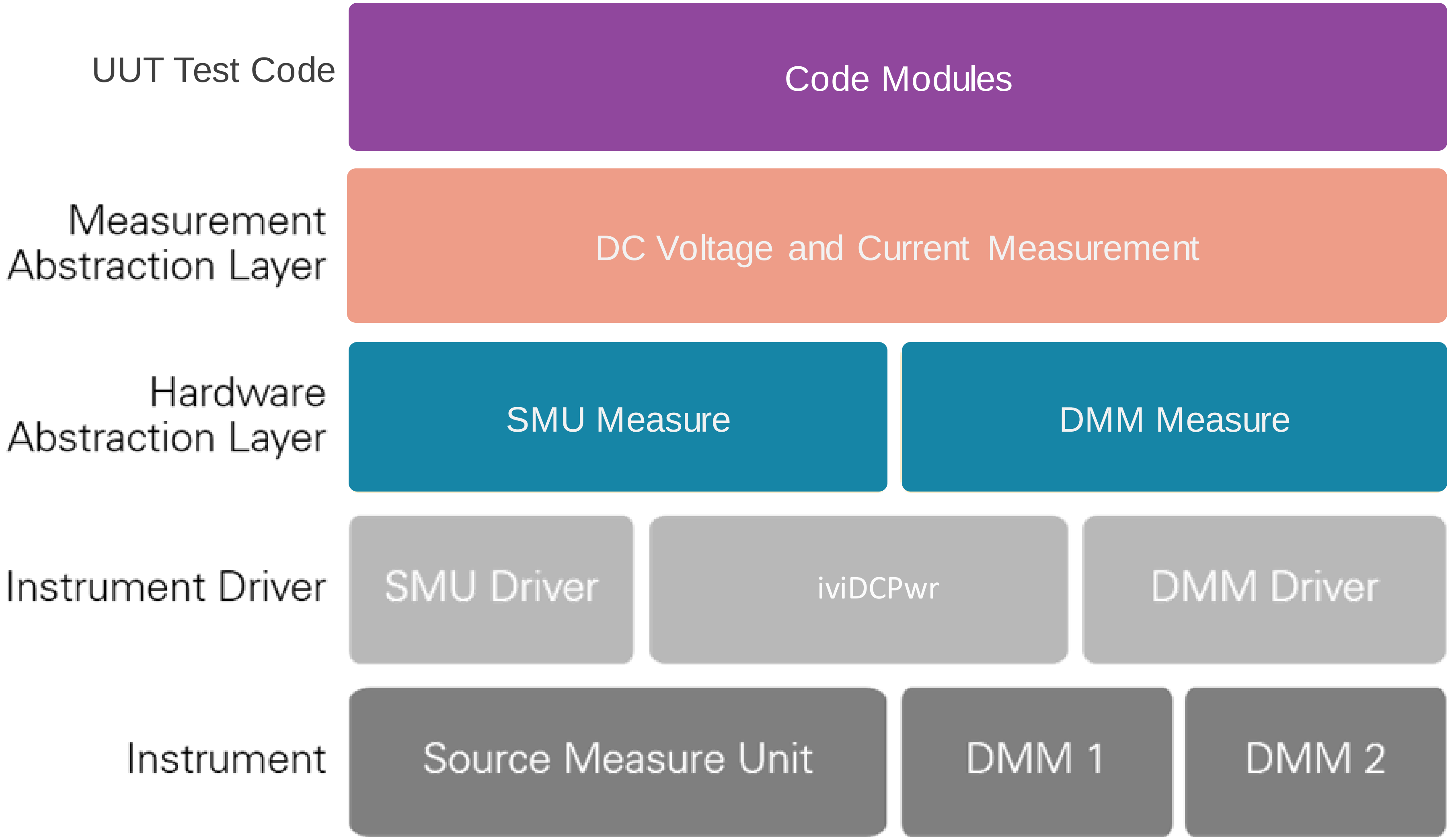
5.- Not making use of commercially available deployment tools

It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Employ Abstraction Layers to Mitigate Risk



UUT Test Code

Code Modules

Measurement Abstraction Layer

DC Voltage and Current Measurement

Hardware Abstraction Layer

SMU Measure

DMM Measure

Instrument Driver

SMU Driver

iviDCPwr

DMM Driver

Instrument

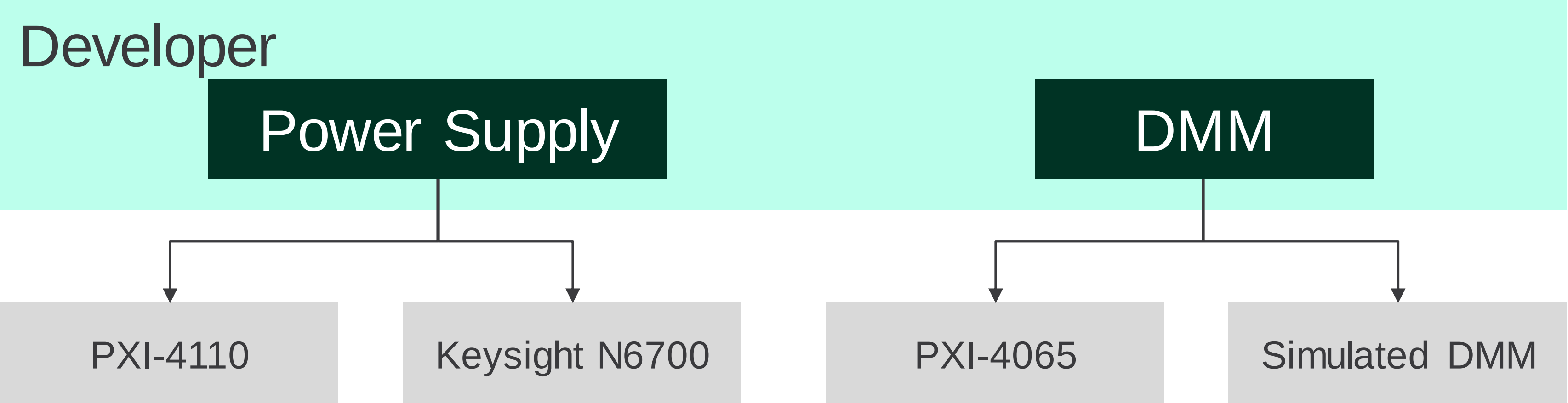
Source Measure Unit

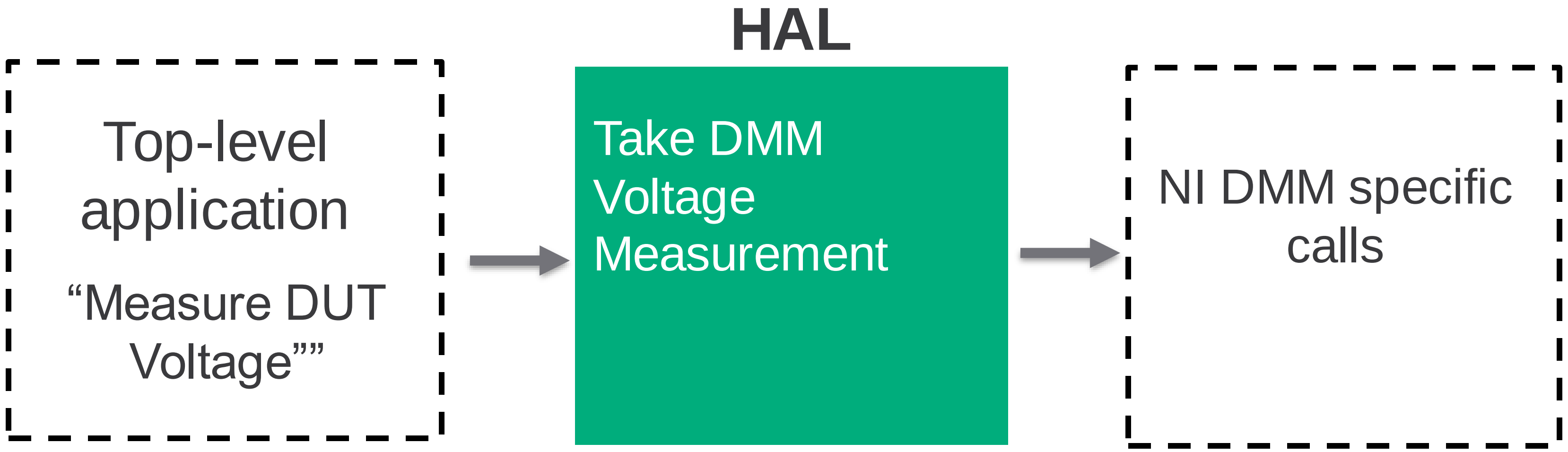
DMM 1

DMM 2

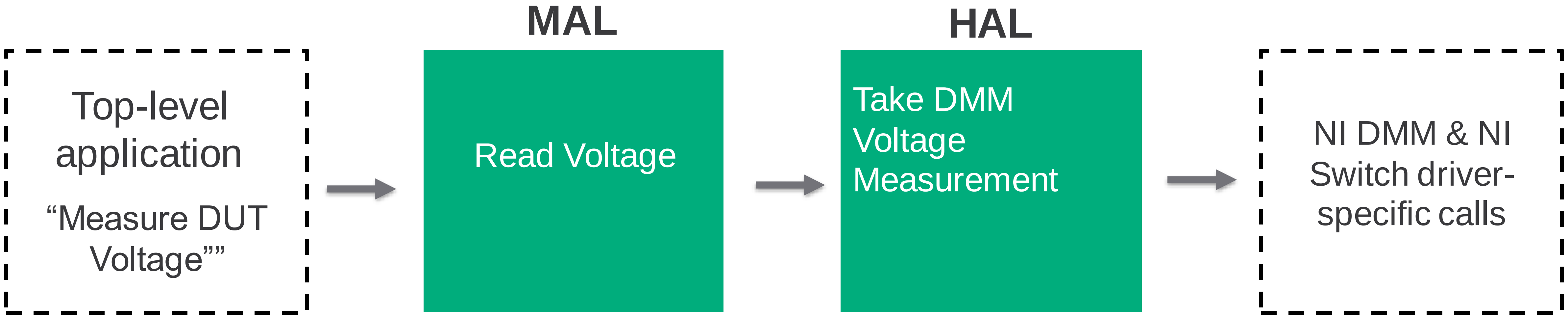
Hardware Abstraction Layer (HAL)

- Identify common functionality.
- What are the functions of a DMM?
- What are the functions of a power supply?

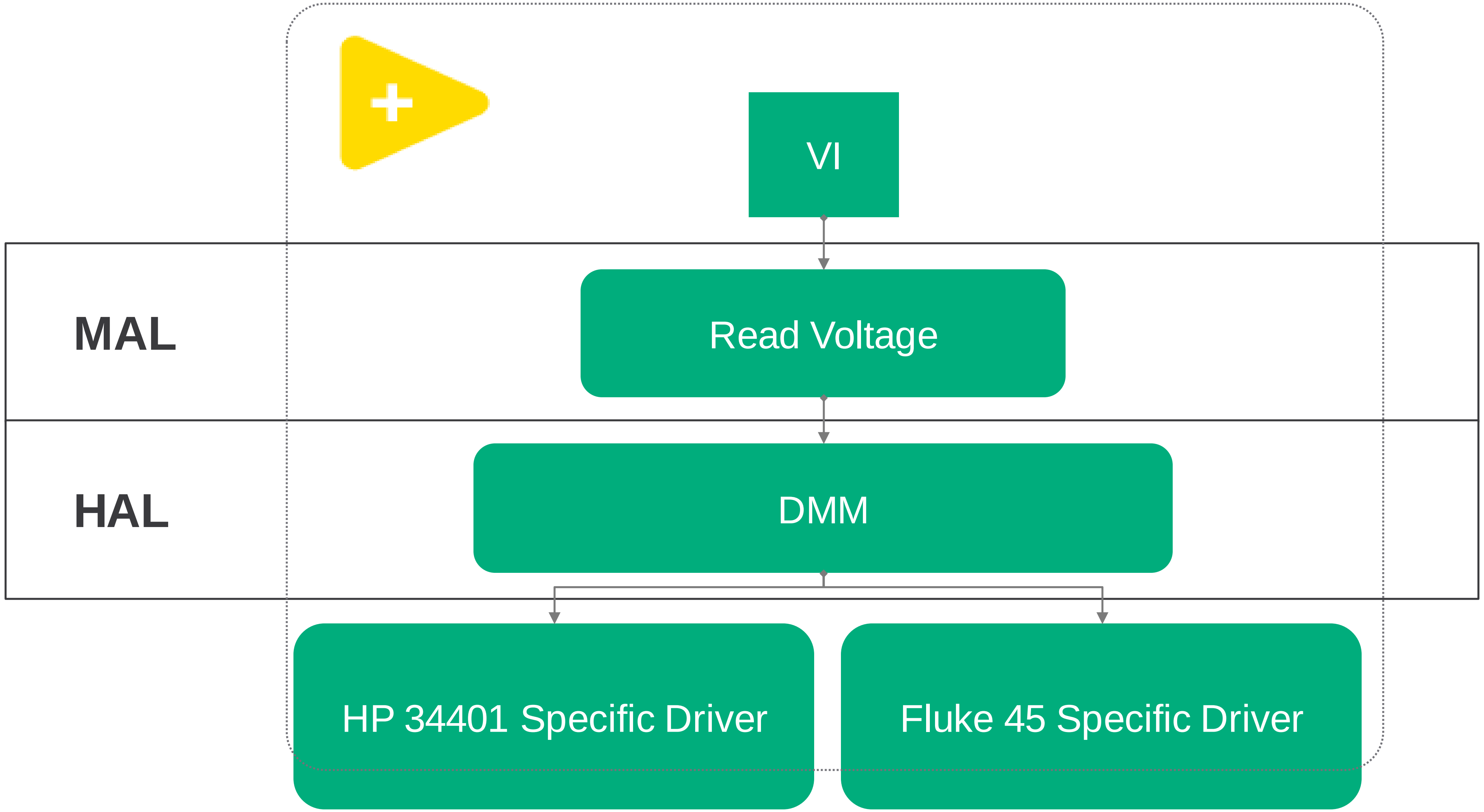




Measurement Abstraction Layer (MAL)


MAL
HAL
Top-level application
“Measure DUT Voltage””
Read Voltage
Take DMM Voltage Measurement
NI DMM & NI Switch driver-specific calls

How do the HAL / MAL fit together?



VI

MAL

Read Voltage

HAL

DMM

HP 34401 Specific Driver

Fluke 45 Specific Driver

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *"If it works… don't touch it!".*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a "file share" only, there is no way of knowing if a remote has a working copy.

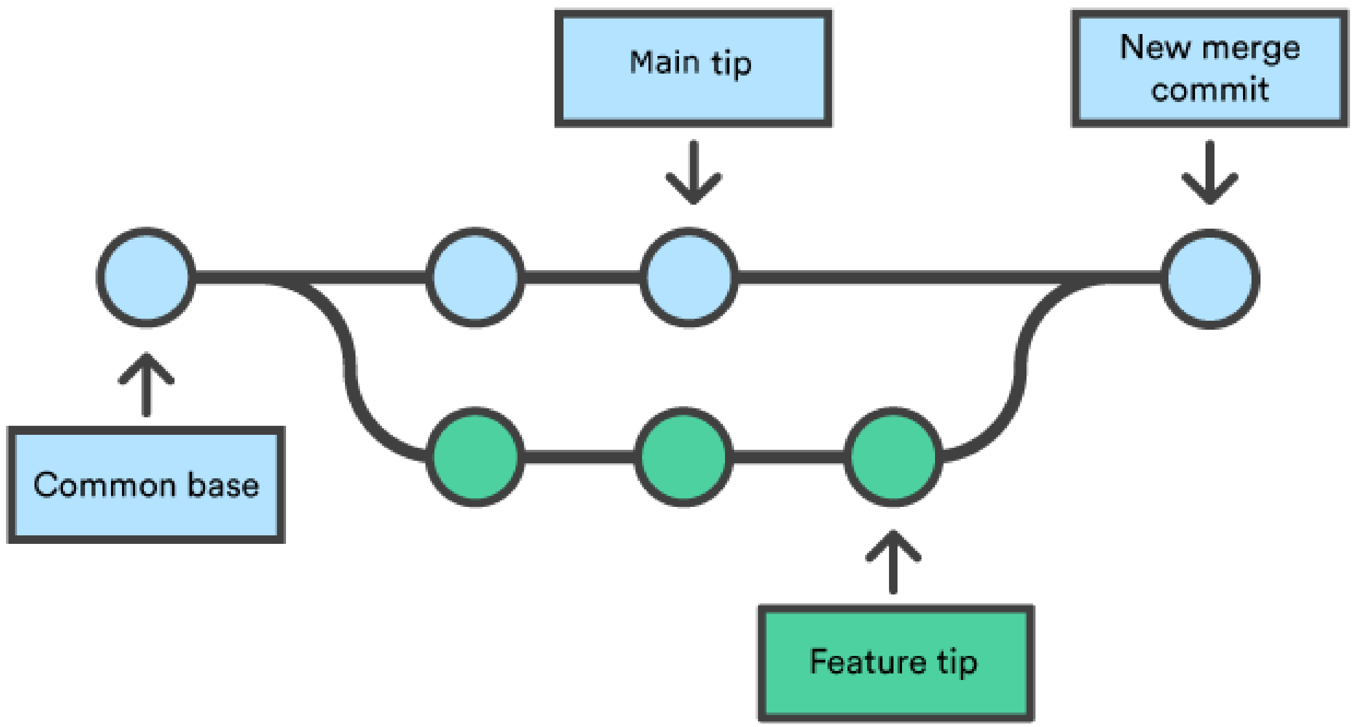
5.- Not making use of commercially available deployment tools

It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Source code control(SCC)



- Code traceability
- Long term efficiency increase
- Simplifies collaboration and workload distribution

The Turing Way Community. This illustration is created by Scriberia with The Turing Way community, used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807

Git hooks and unit testing

- Unit testing is a tool for automating testing your source code against a specific condition.
- Unit testing was added to source code control. If your code fails the unit test, you can't commit until you pass it.
- In this case, ONE simple unit test made collaboration seamless: **IS THERE A BROKEN ARROW?**
- More tests were added later, that enforced good practices by adding VI Analyzer test with specific good practices that want to be enforced on the source code.
- This approach comes with its risks:
  - **Too much unit testing** would make committing a new feature very restrictive, adding overhead.
  - **Not enough unit testing** would give the false impression that you didn’t broke any rules.

But, when properly defined, this allows for architects to enforce project rules without having to make code review for each of the commits.

Bad code that gets committed… stays bad. - Chris Roebuck

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *“If it works… don’t touch it!”.*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a “file share” only, there is no way of knowing if a remote has a working copy.

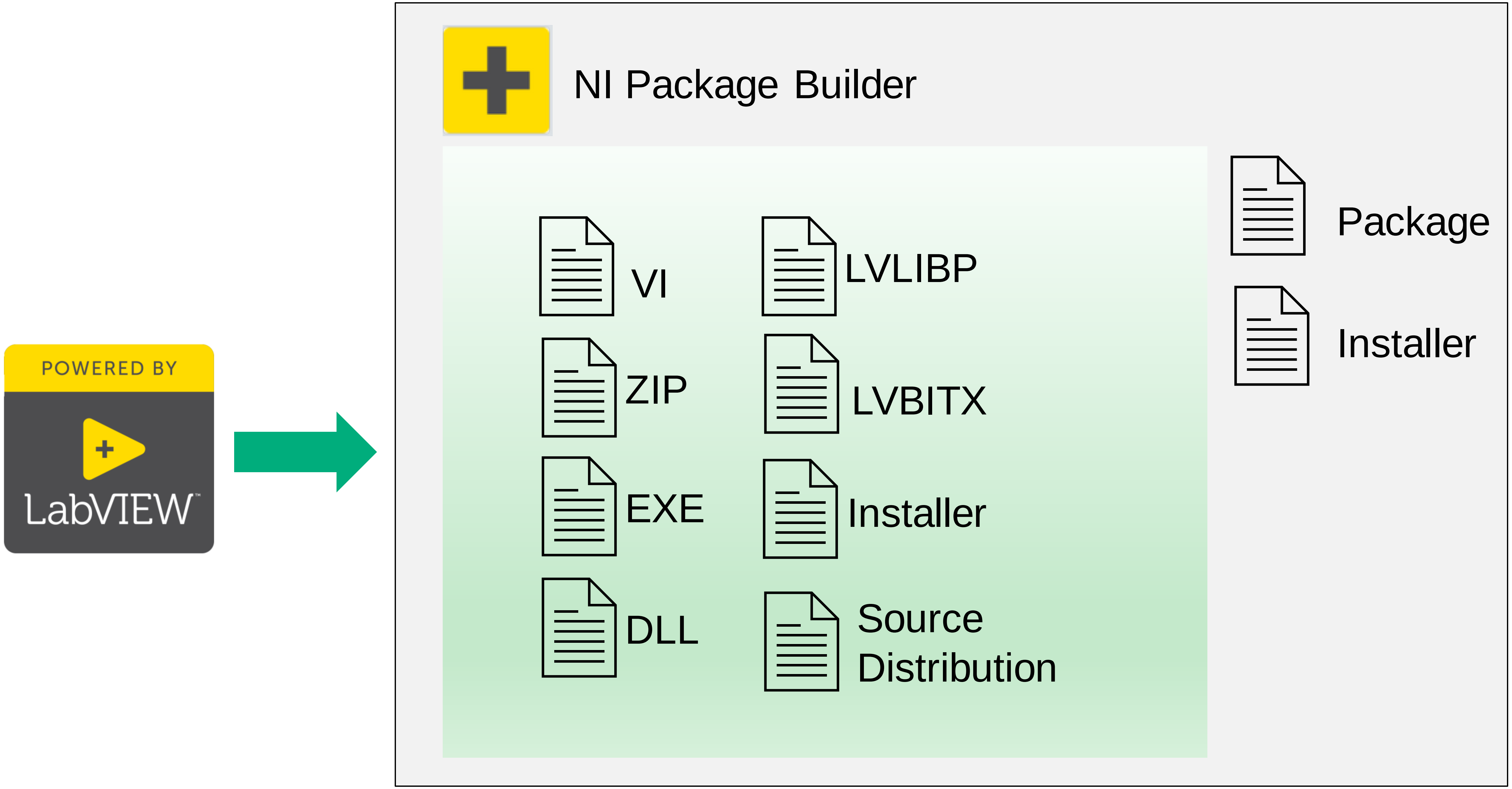
5.- Not making use of commercially available deployment tools

It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Deployment



NI Package Builder

VI

LVLIBP

ZIP

LVBITX

EXE

Installer

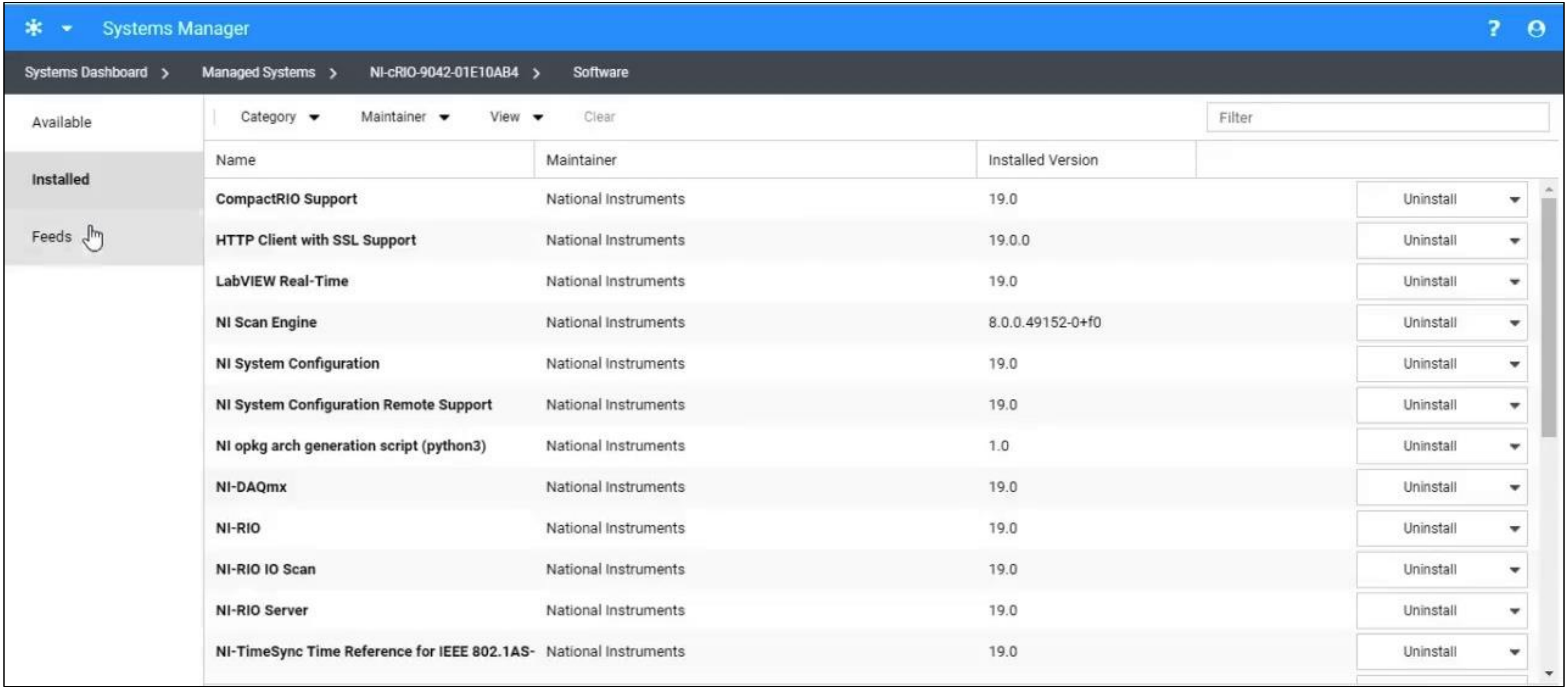
DLL

Source Distribution

Package

Installer

Deployment



Name	Maintainer	Installed Version	
CompactRIO Support	National Instruments	19.0	Uninstall
HTTP Client with SSL Support	National Instruments	19.0.0	Uninstall
LabVIEW Real-Time	National Instruments	19.0	Uninstall
NI Scan Engine	National Instruments	8.0.0.49152-0+f0	Uninstall
NI System Configuration	National Instruments	19.0	Uninstall
NI System Configuration Remote Support	National Instruments	19.0	Uninstall
NI opkg arch generation script (python3)	National Instruments	1.0	Uninstall
NI-DAQmx	National Instruments	19.0	Uninstall
NI-RIO	National Instruments	19.0	Uninstall
NI-RIO IO Scan	National Instruments	19.0	Uninstall
NI-RIO Server	National Instruments	19.0	Uninstall
NI-TimeSync Time Reference for IEEE 802.1AS-	National Instruments	19.0	Uninstall

Case Study 1: Global validation lab

1.- Coupling and technical debt

Validation engineers take the faster path to solve a problem due to an unofficial company wide approach of solving issues reactively. *“If it works… don’t touch it!”.*

2.- Not having a standardized way of collecting results

Result collection is implemented differently across sites, results are then shared by attaching them to an email. Serial number of the DUT must be manually typed at the beginning of a test.

3.- Lack of Hardware Abstraction

When trying to replicate results at a different validation lab, the test sequence now needs to be rewritten since it was made around a specific instrument not available at that specific validation lab.

4.- No strategy when using source code control

Source code is used as a “file share” only, there is no way of knowing if a remote has a working copy.

5.- Not making use of commercially available deployment tools

It is known that setting up a workstation can only be made by the validation engineer who has learned the most about NI Software, which are no more than a couple per site.

6.- Not having a standardized error handling strategy

Test sequences are constantly stopping the test due to mistakes that were made by the developer.

Error handling strategy

- Things happen such as error codes, timeouts, infinite loops. There is no utopia and it i should be expected that errors will occur.

- In this case, we needed assurances that if an error occurs, there would be a mechanism to protect critical components of the system.

- By making use of the already existing deployment infrastructure made on previous stage of this project, we were able to deploy TestStand files that are components of the custom sequential process model.

- This validation lab no longer has validation boards that are one of a kind, soaked in water due to an instrument that injects cold air being left on since it had the misfortune of an error happening in the middle of injecting cold air.

Which framework is the best?

Actor Framework

Case Study

Lidar PCBA Test system

Challenge : 1 .- Multiple Board Testing + 2.- Different Final Customer Customization

Test Rack

PXI Technology

Mass Interconnection Harness

(Sectoring signal model)

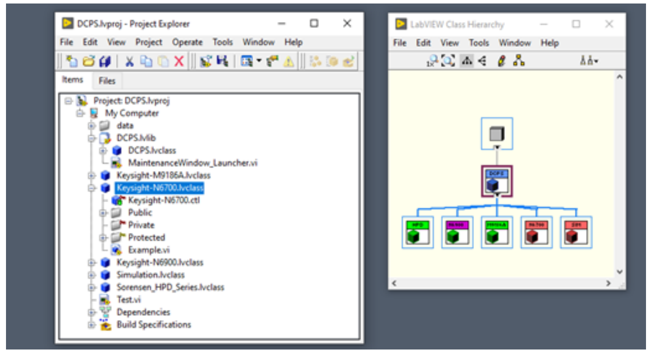
Fixture Wireless

PCBA interface - Mirror signal contacts

Hardware Architecture

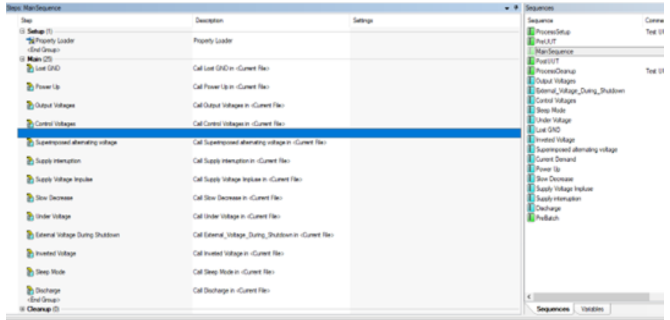
Case Study

Development Environment


DCPS.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
LabVIEW Class Hierarchy
File Edit View Tools Window Help

Project: DCPS.lvproj
- My Computer
  - data
  - DCPS.lvlib
    - DCPS.lvclass
    - MaintenanceWindow_Launcher.vi
  - Keysight-M9186A.lvclass
  - Keysight-N6700.lvclass
    - Keysight-N6700.ctl
    - Public
    - Private
    - Protected
    - Example.vi
  - Keysight-N6900.lvclass
  - Simulation.lvclass
  - Sorensen_HPD_Series.lvclass
  - Test.vi
  - Dependencies
  - Build Specifications

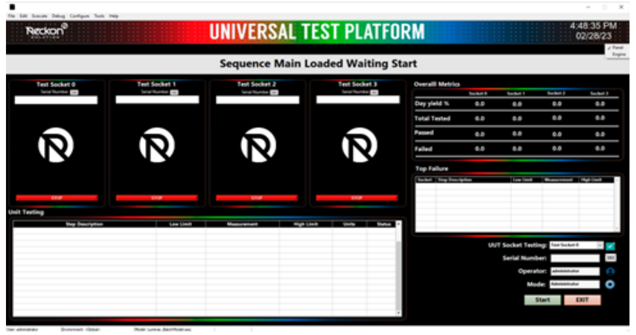
NI LabVIEW
Object Oriented Programming

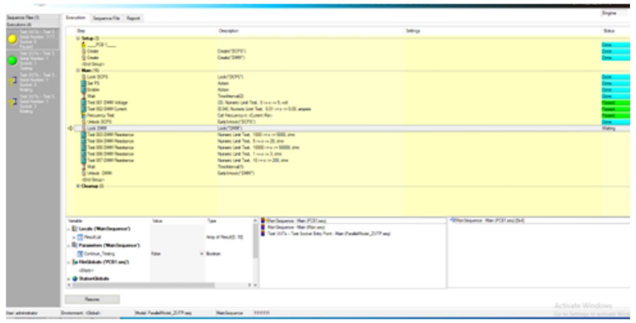


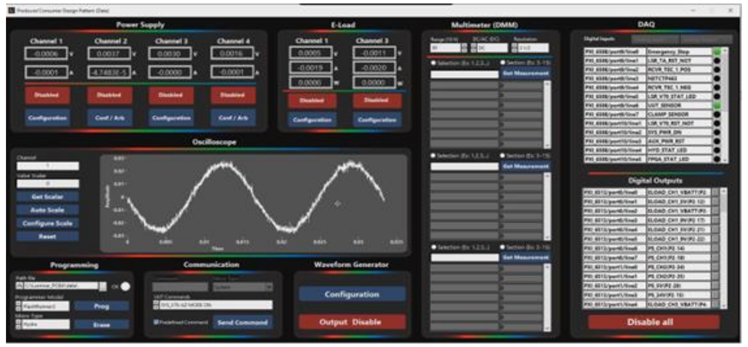
NI TestStand
Sequence Deployment

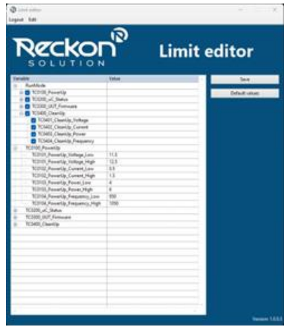
Run-Time Environment

NI LabVIEW
User Interface









NI LabVIEW
Maintenance Interface

NI LabVIEW
Product Configuration Interface

Software Architecture

2024

CONNECT