What the HAL?

Using a Hardware Abstraction Layer for automated test

MEET THE PRESENTERS

ABOUT US

HAL ARCHITECT

Tom Kantner

Tom.kantner@testeract.com

BS Aerospace Eng, UCLA

5+ years Automated Test

CLD, CTD

HIL, PLC Expert

PRESIDENT/FOUNDER

Sam Roundy –

sam@testeract.com

BS Computer Science, Utah

20+ years Automated Test

CLA, CTA, LabVIEW Champion

ABOUT TESTERACT

Specialize in Automated Test Software and Test Systems

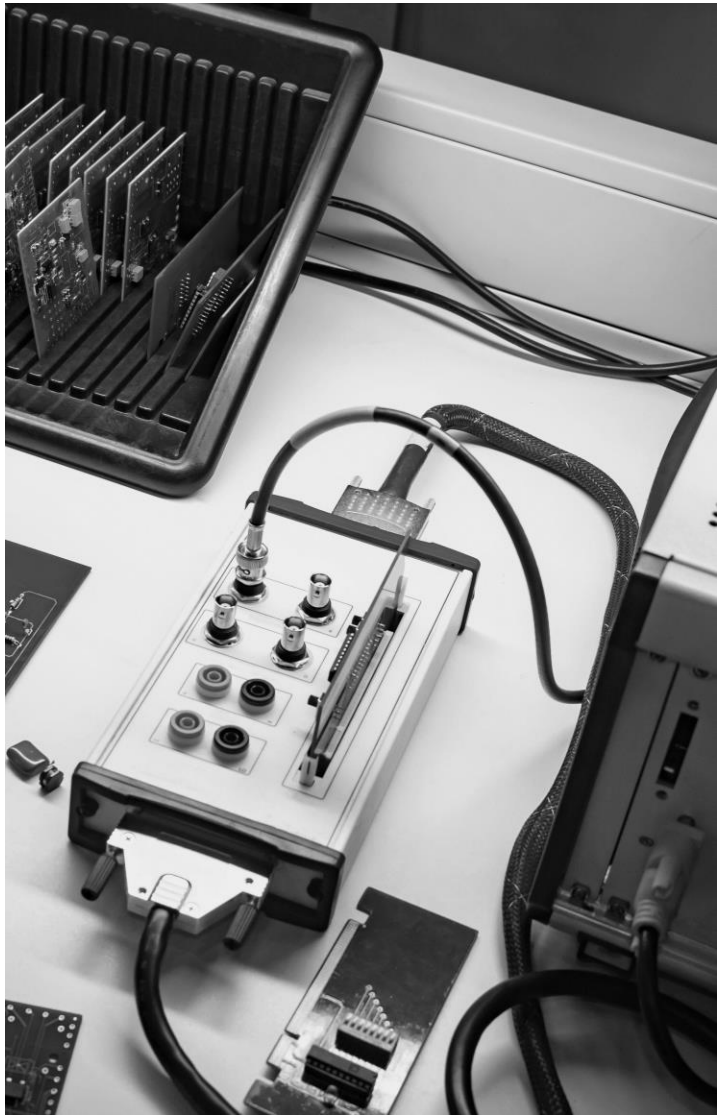
40+ Engineers

(LabVIEW, TestStand, Python, C#, Databases, Test Hardware)

Founded: 2015

TestPoint Framework

contact@testeract.com



In this Presentation

AUTOMATED TEST SOFTWARE STACK

INSTRUMENT CONTROL

SWAPPING INSTRUMENTS

OTHER ADVANTAGES

DEMOS AND THINGS

DISCUSSION

Testeract

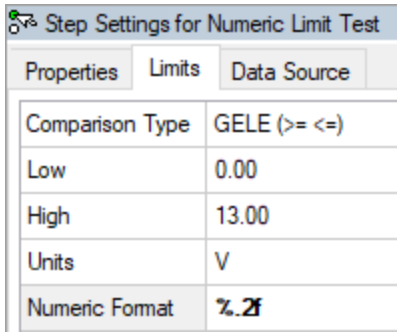
Understanding the SW Stack

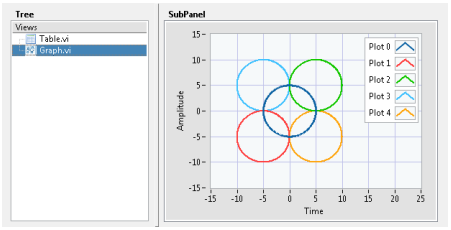
Testeract

Every Hardware Application Has These 4 Components!

Acquire	Analyze	Present	Govern
• Talk to Hardware • Protocols • Buses	• Limits • Measurements • Pass/Fail • Sequences	• User Experience • Debug Tools • Data (DB, Reports) • Diagnostics • Configuration	• Software Management • Training • Roles • Processes (Workflow)

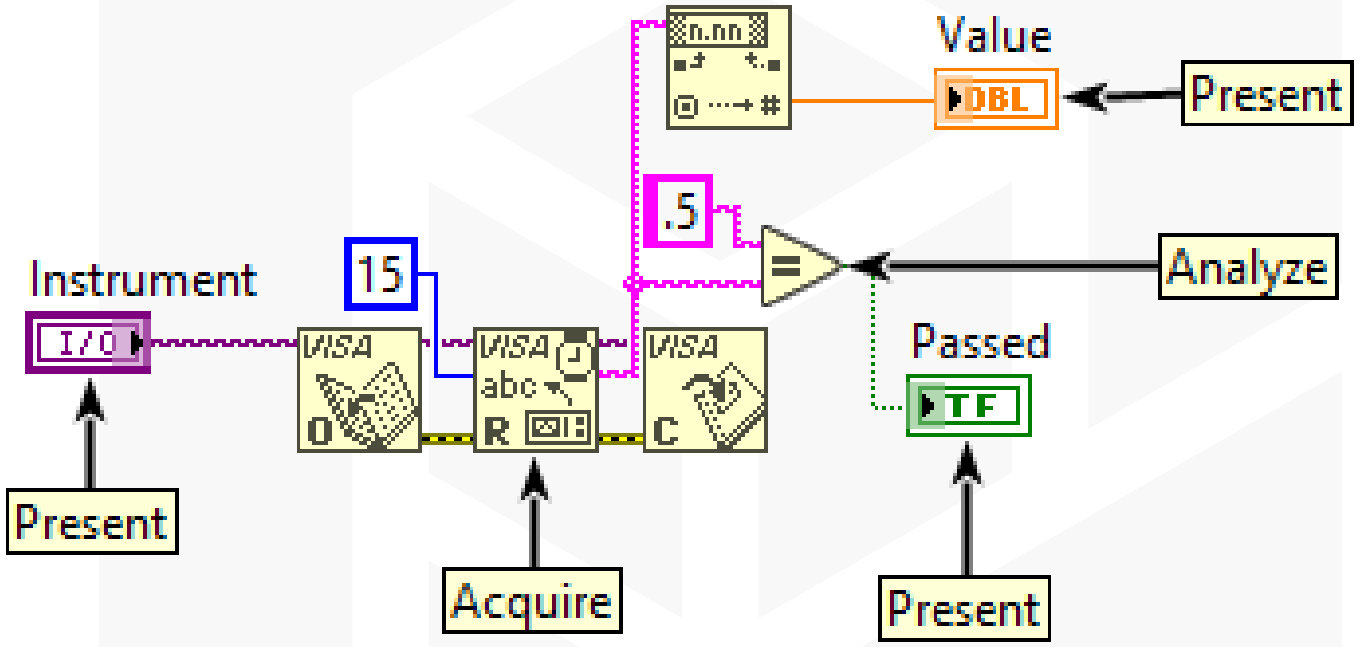
Examples





testeract
-ts-x64-h
ml-ups_3
.4.0-0_...

Simple Example – All 3 Major Components



What is a HAL?

Testeract

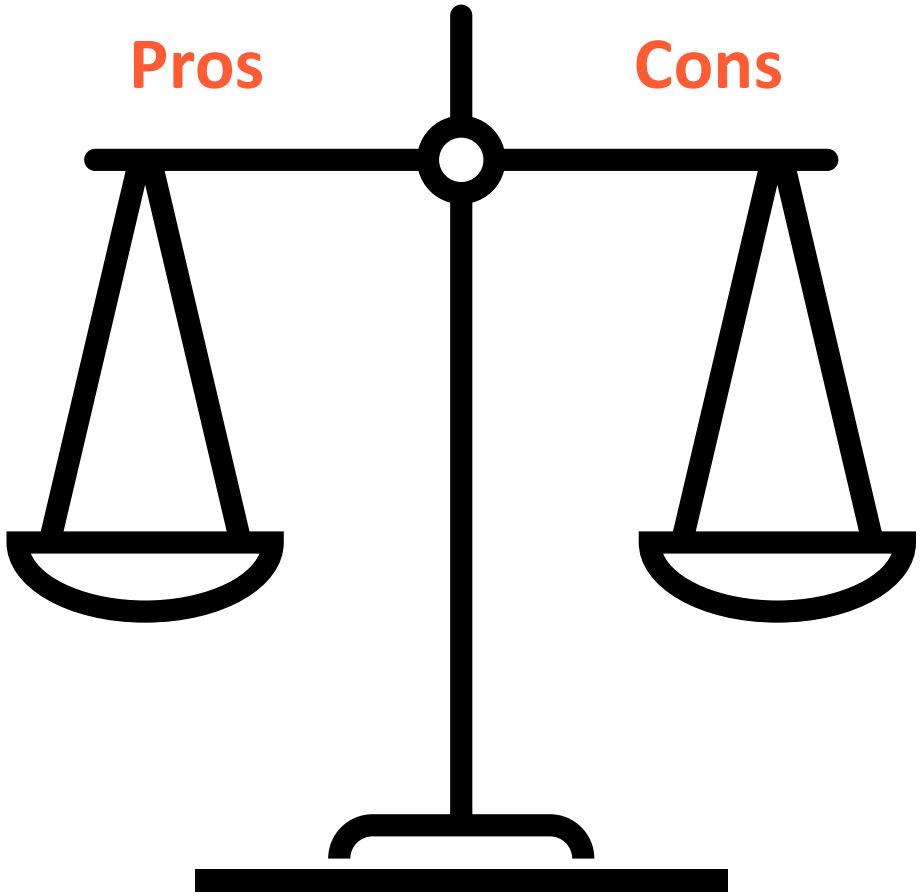
Every Hardware Application Has These 4 Components!

Acquire **Analyze** **Present** **Govern**

- Stands for Hardware Abstraction Layer
- Layer in the SW stack that directly interfaces with the hardware
  - **Abstracts** the User from the Low-Level communication hardware details
- Encapsulates the information needed to interface with the instrumentation for easy decoupling from rest of SW stack
  - Each device is responsible for its data and only its data
  - API provides easy access for controlling and changing state

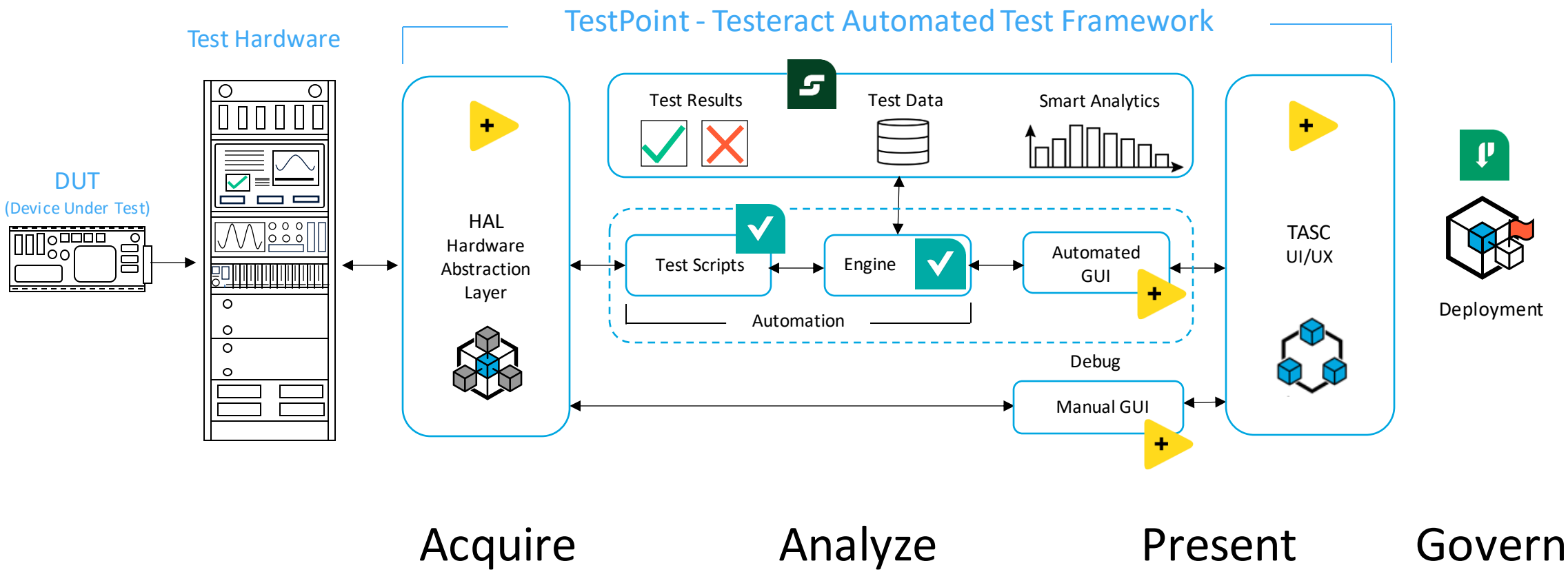
Do I Need a HAL?

- Without it new instruments are harder to add later (technical debt)
- Parallelizing Development
- Protects against obsolescence



- Higher up front development cost
- “I’m only developing for one set of hardware”
- “The required hardware will never change”

Testeract TestPoint Framework



Goal of a HAL – Maximize Reuse

Swap

Unchanged

Keysight

?

Sorensen

Driver
Part that talks to Instrument or UUT

HAL
Hardware Abstraction Layer

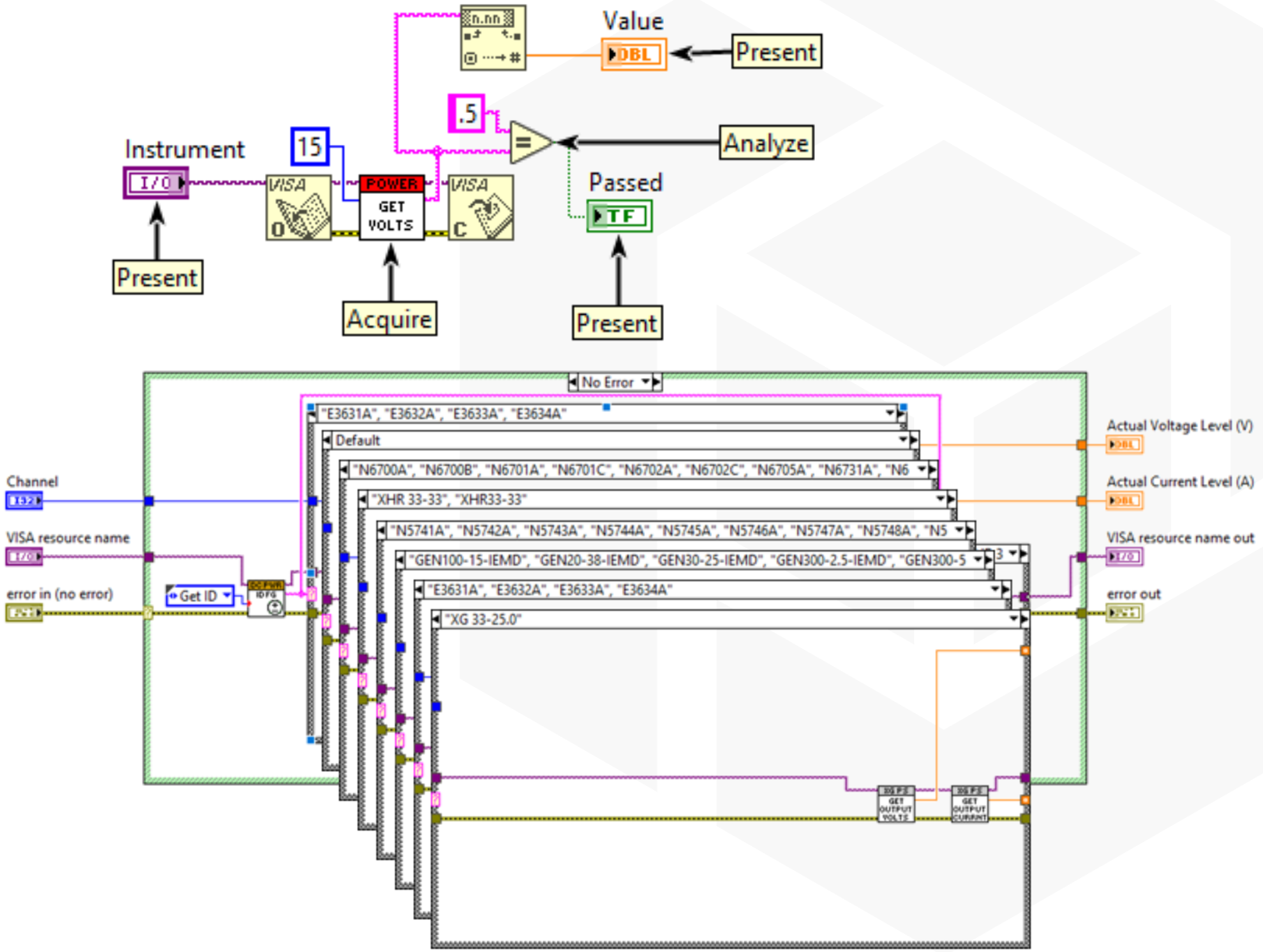
Analyze

Present

Govern

Acquire

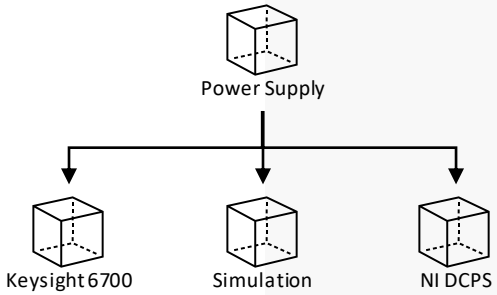
Simple HAL Example

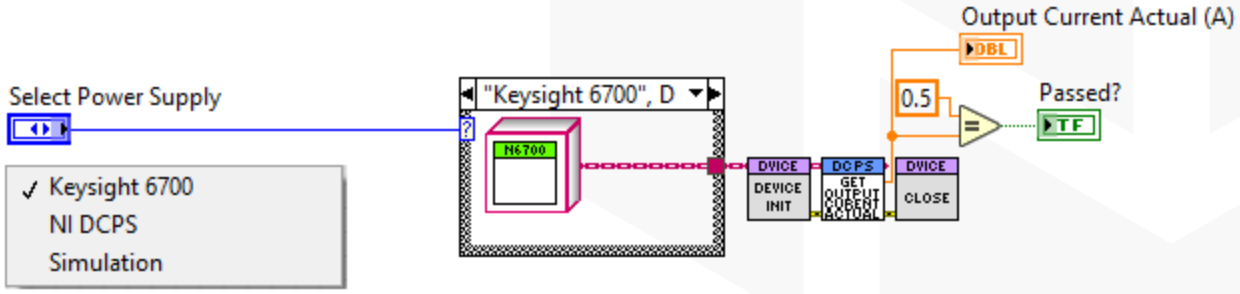


Problems:

- Adding code forces you to touch the API
- Rebuild every time instrument is added
- Regression Testing
- Limited to VISA

Object Oriented Solves those Problems





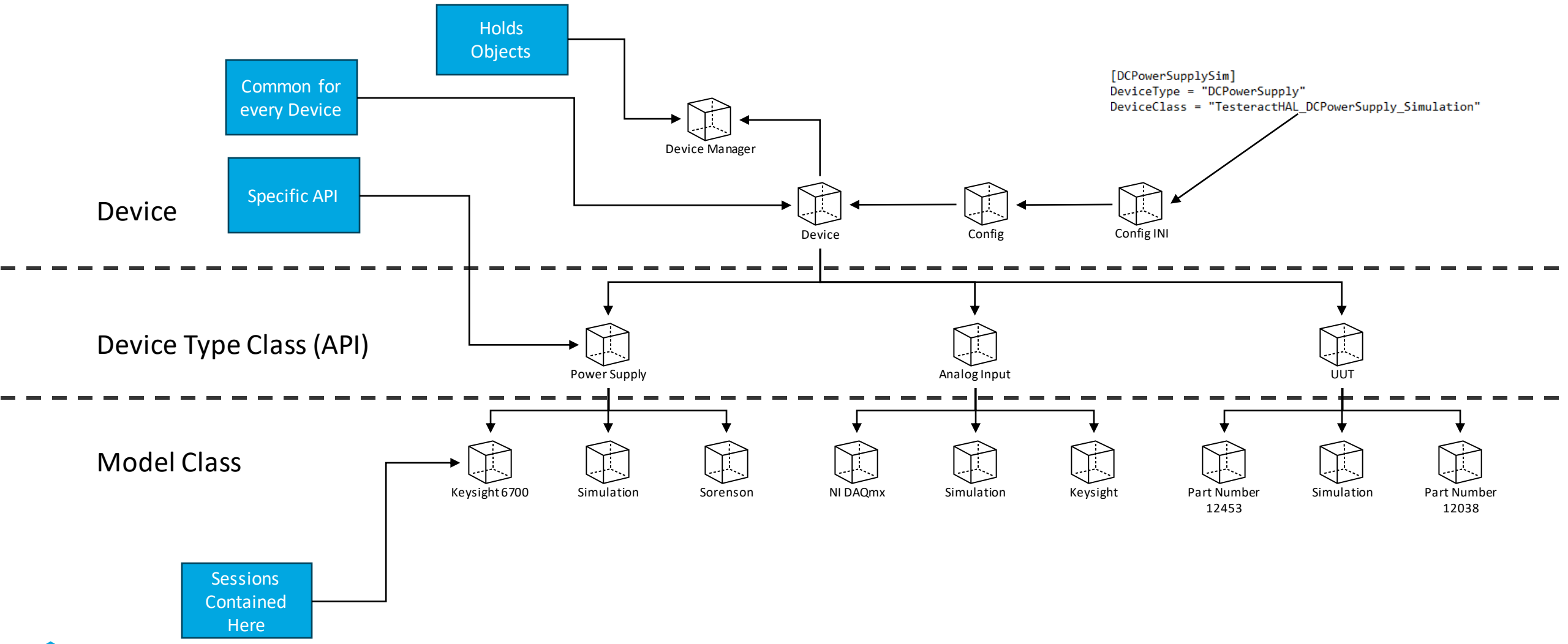
Problems Solved:

- Adding code forces you to touch the API
  - **No longer needed**
- Rebuild every time instrument is added
  - **Depending on how you do it, can go away**
- Regression Testing
  - **Only test the stuff you added**
- Limited to VISA
  - **Bus and protocol are encapsulated in object**

Potential HAL Features/Requirements

Feature	Notes
Swapping	Ability to swap like instruments without touching the code
Support Any Device	Ability to work with any bus, instrument, connection, etc.
Simulation	Ability to simulate entire system
Logging	Log all HAL interactions without impacting speed
Tools	Standardization allows for many tools to be created (HAL Debug Tool)
Record and Playback	Capture HAL calls and play them back
Shared Connections	Treat multi-personality instruments as different instruments
UUT Within The HAL	Trust me- you'll want all the HAL goodies for your UUT

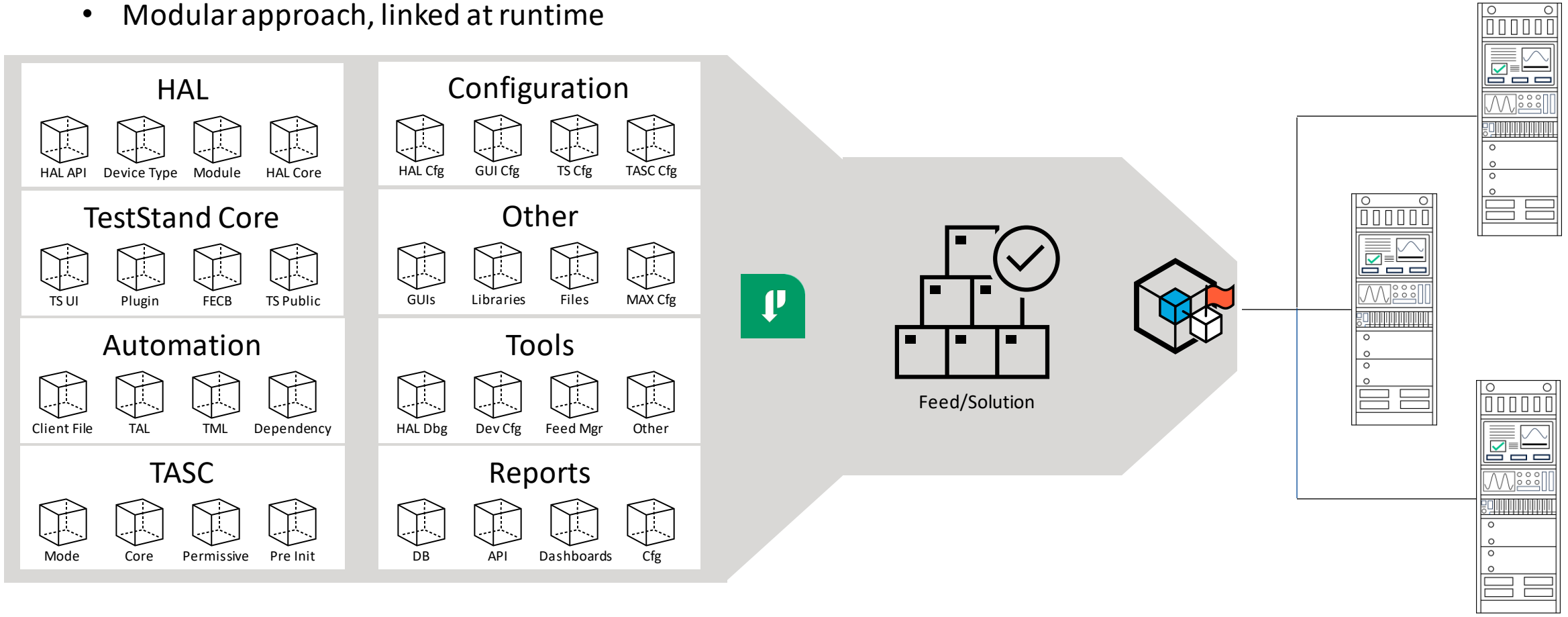
HAL Overview



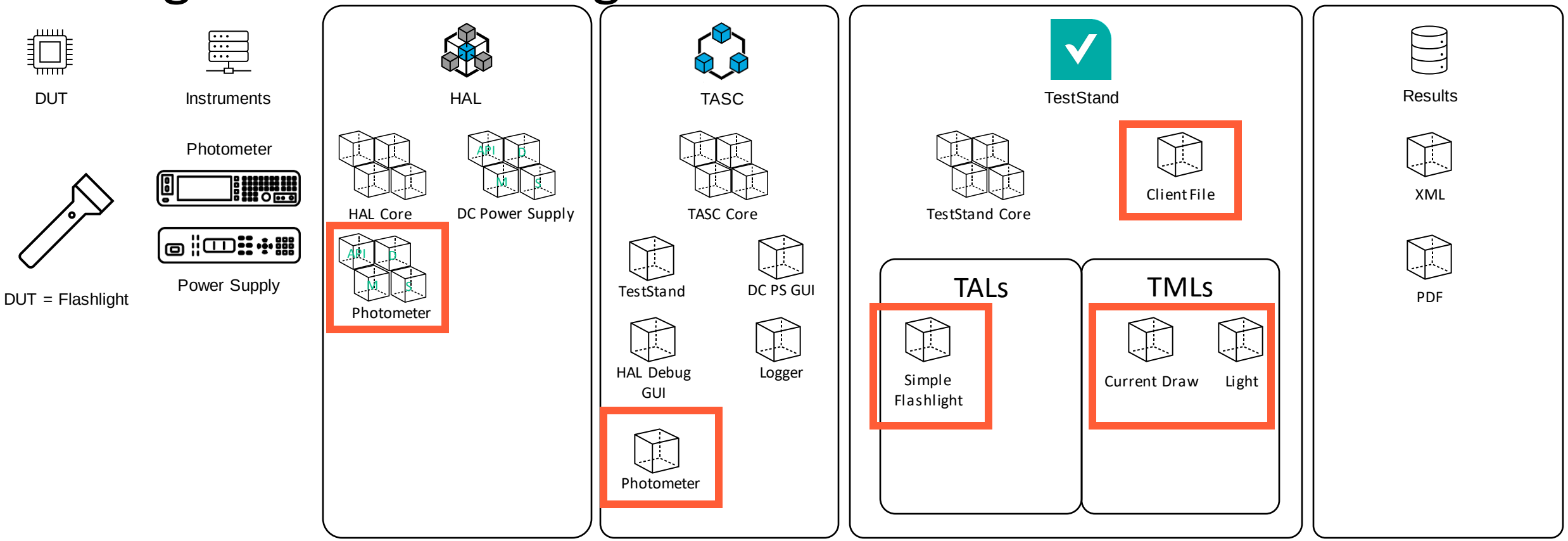
Testeract

Deployment

- Deployed PC only consumes packages it needs
- PPLs used to distribute code
  - Modular approach, linked at runtime



Flashlight Tester - Packages

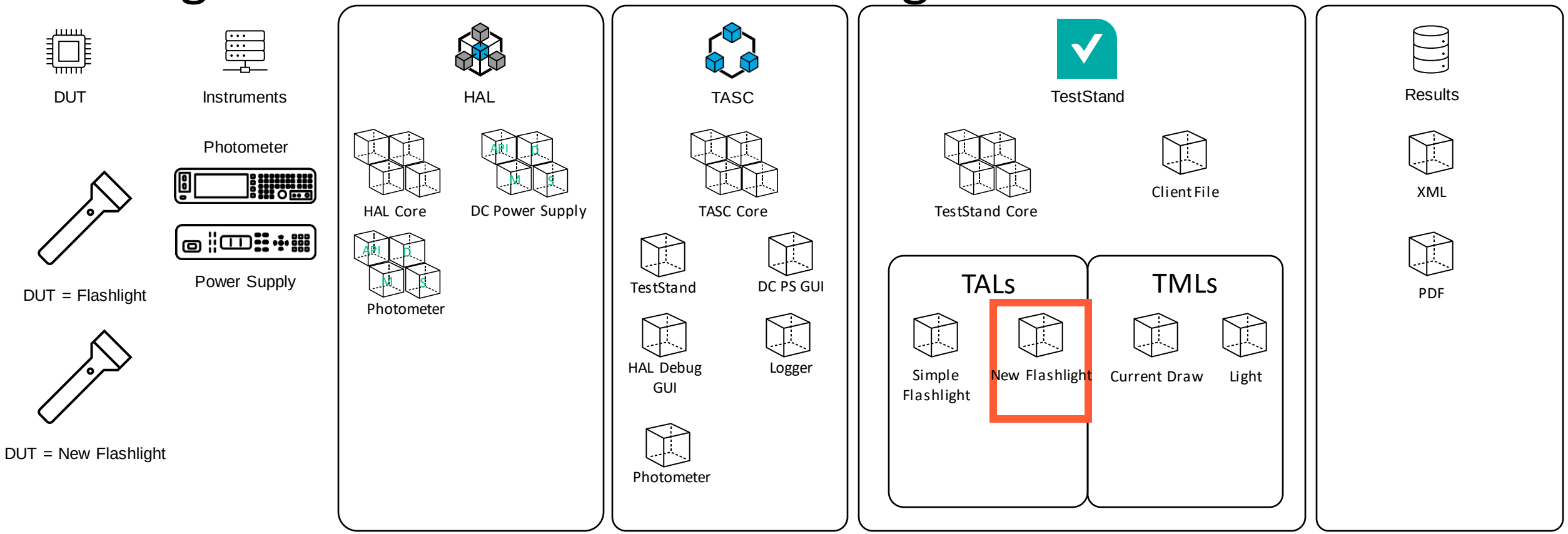


=

- Packages we had to create or modify for this test station
- Can be reused in the future

Testeract

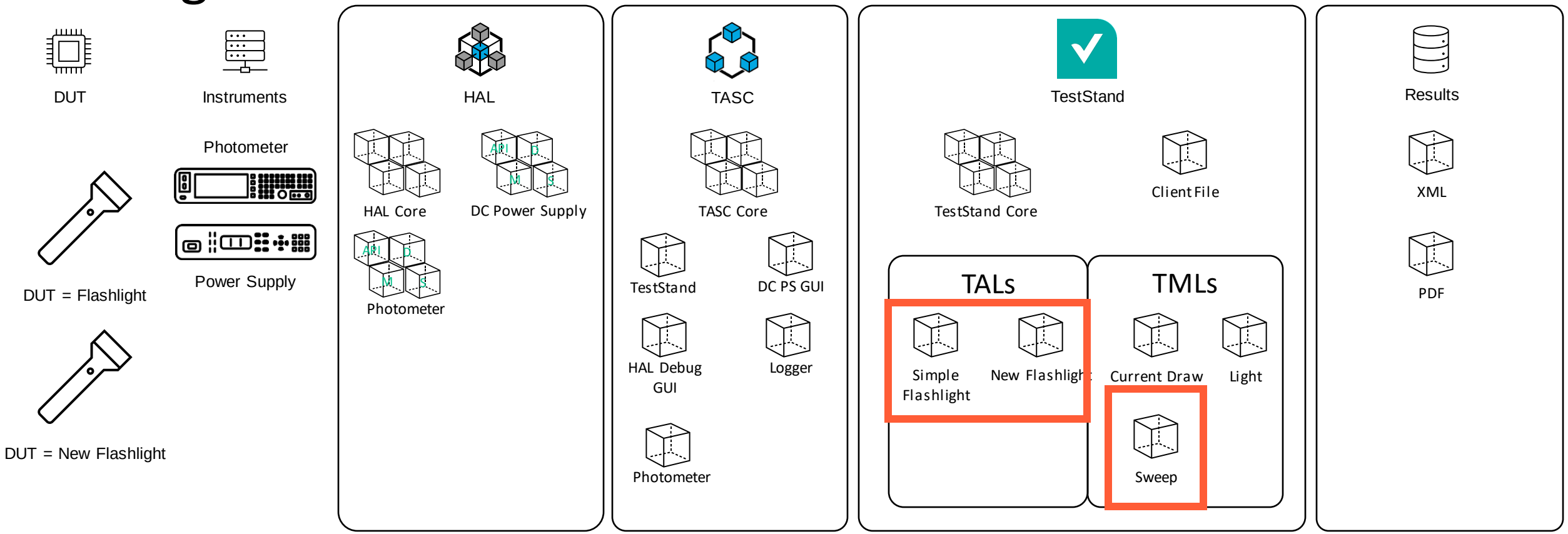
Flashlight Tester – Add a new Flashlight



=

- Just add a new TAL file

Flashlight Tester – Add a new Test Module

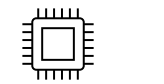


=

- Add new TML
- Modify TALs to call TMLs

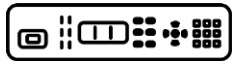
Testeract

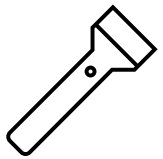
Flashlight Tester – Add DUT Com


DUT

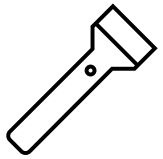

Instruments

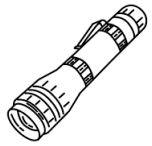
Photometer


Power Supply


Flashlight


I2C Bus


New Flashlight


Tri State Flashlight


HAL

HAL Core

DC Power Supply

Photometer

Tri State Flashlight


TASC

TASC Core

TestStand

DC PS GUI

HAL Debug GUI

Logger

Photometer


TestStand

TestStand Core

Client File

TALs

Simple Flashlight

New Flashlight

Tri State Flashlight

TMLs

Current Draw

Light

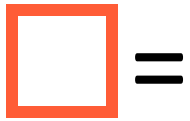
Sweep

Tri State Light


Results

XML

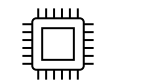
PDF


=

- Add new DUT HAL
- Add TMLs and TALs as needed
- Can add TASC GUIs

Testeract

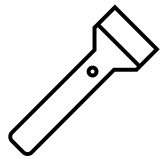
Flashlight Tester – Thermal Testing


DUT

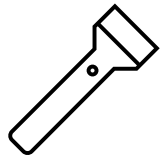

Instruments

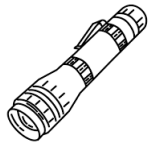
Photometer


Power Supply

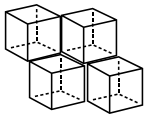

Flashlight

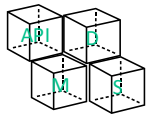

I2C Bus


New Flashlight

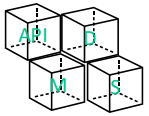

Tri State Flashlight


HAL

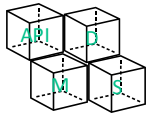

HAL Core



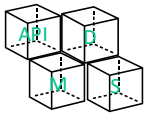
DC Power Supply



Photometer

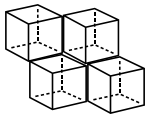


Tri State Flashlight



Temp Chamber


TASC


TASC Core


TestStand

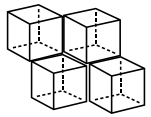

DC PS GUI


HAL Debug GUI


Logger


Photometer


TestStand


TestStand Core


Thermal Client File

TALs


Simple Flashlight


New Flashlight


Tri State Flashlight

TMLs


Current Draw


Light

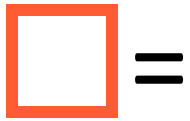

Sweep


Tri State Light


Results


XML


PDF


=

- Add new Client File
- Add Thermal Drivers (already in library)
- Might need slight changes to TMLs/TALs for batch testing

Testeract

Demo

Configure Instruments

TASC

Logging and Record & Playback

Software Management Toolkit (SMT)

Testeract

Q+A

Testeract

Contact Us!

SUPPORT

support@testeract.com

HALSupport@testeract.com

Email anytime for help with any issues for access to an engineer.

TESTERACT

contact@testeract.com

For any other issues

SUPPORT