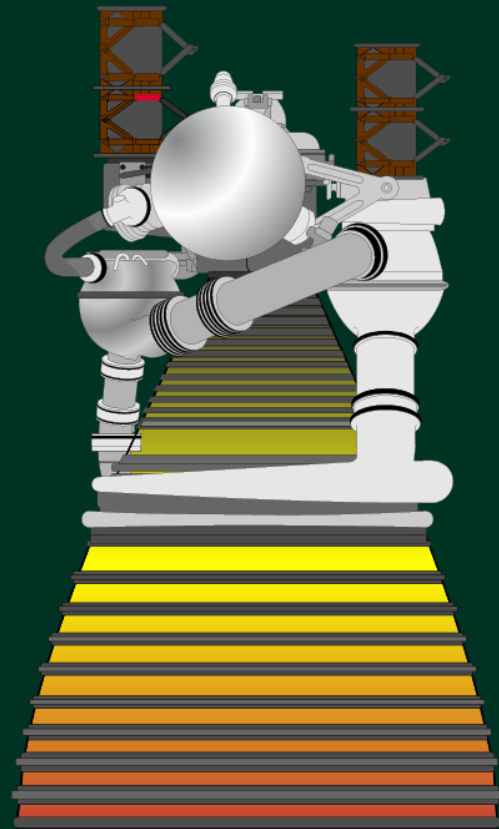
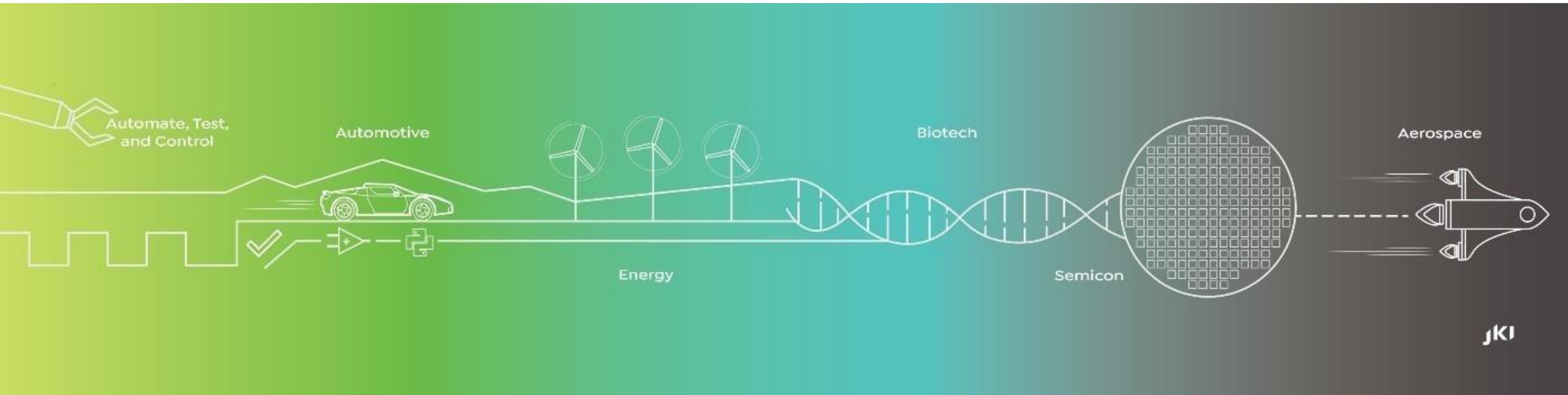


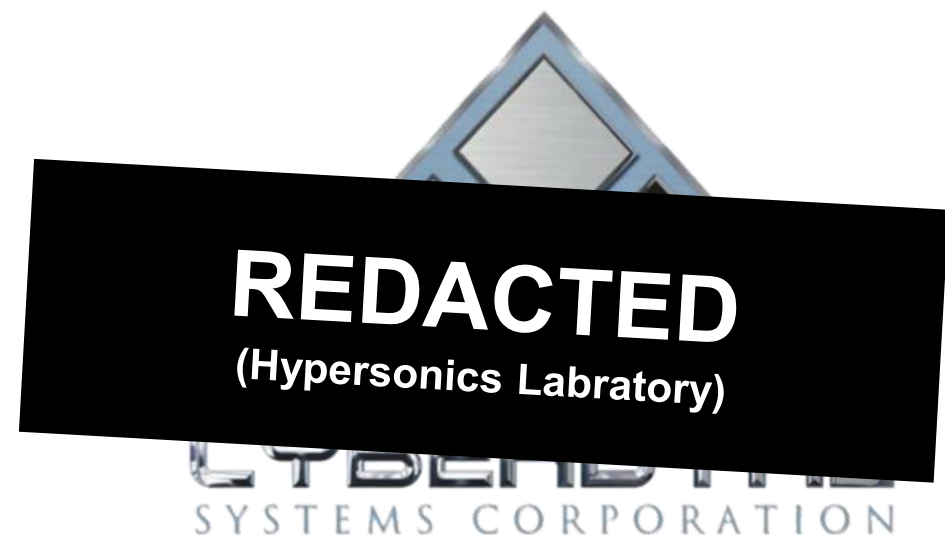
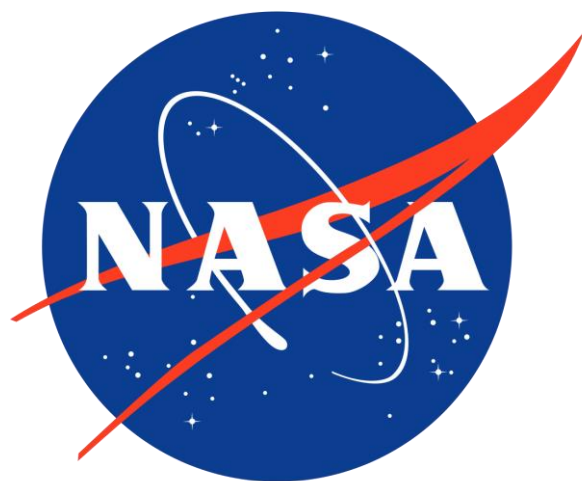
# Developing Robust Rocket Test Software with LabVIEW

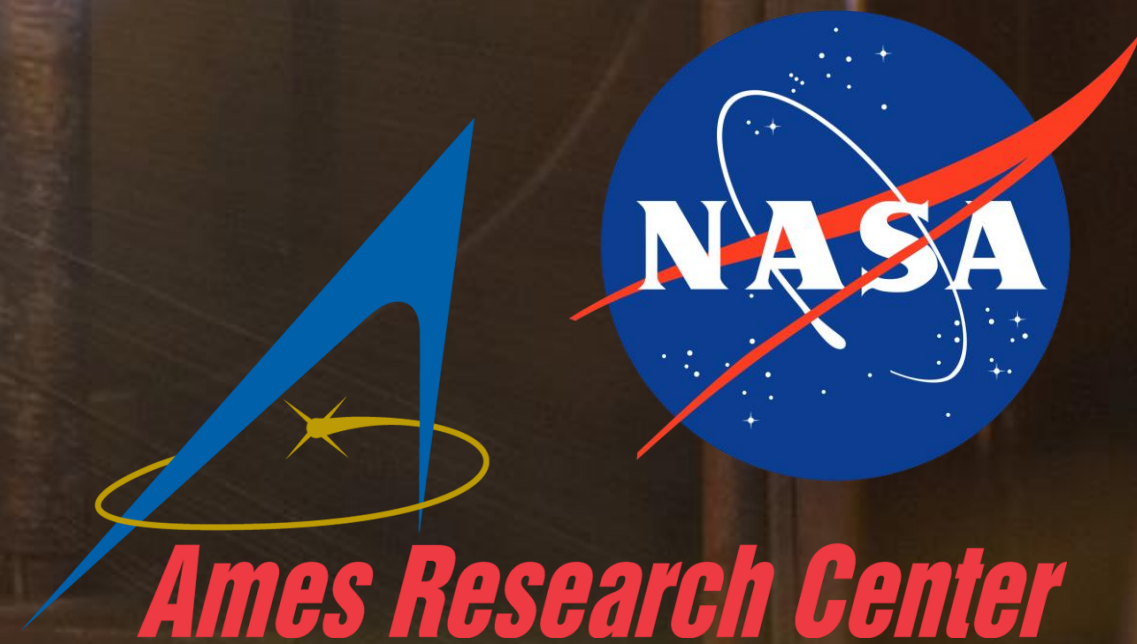
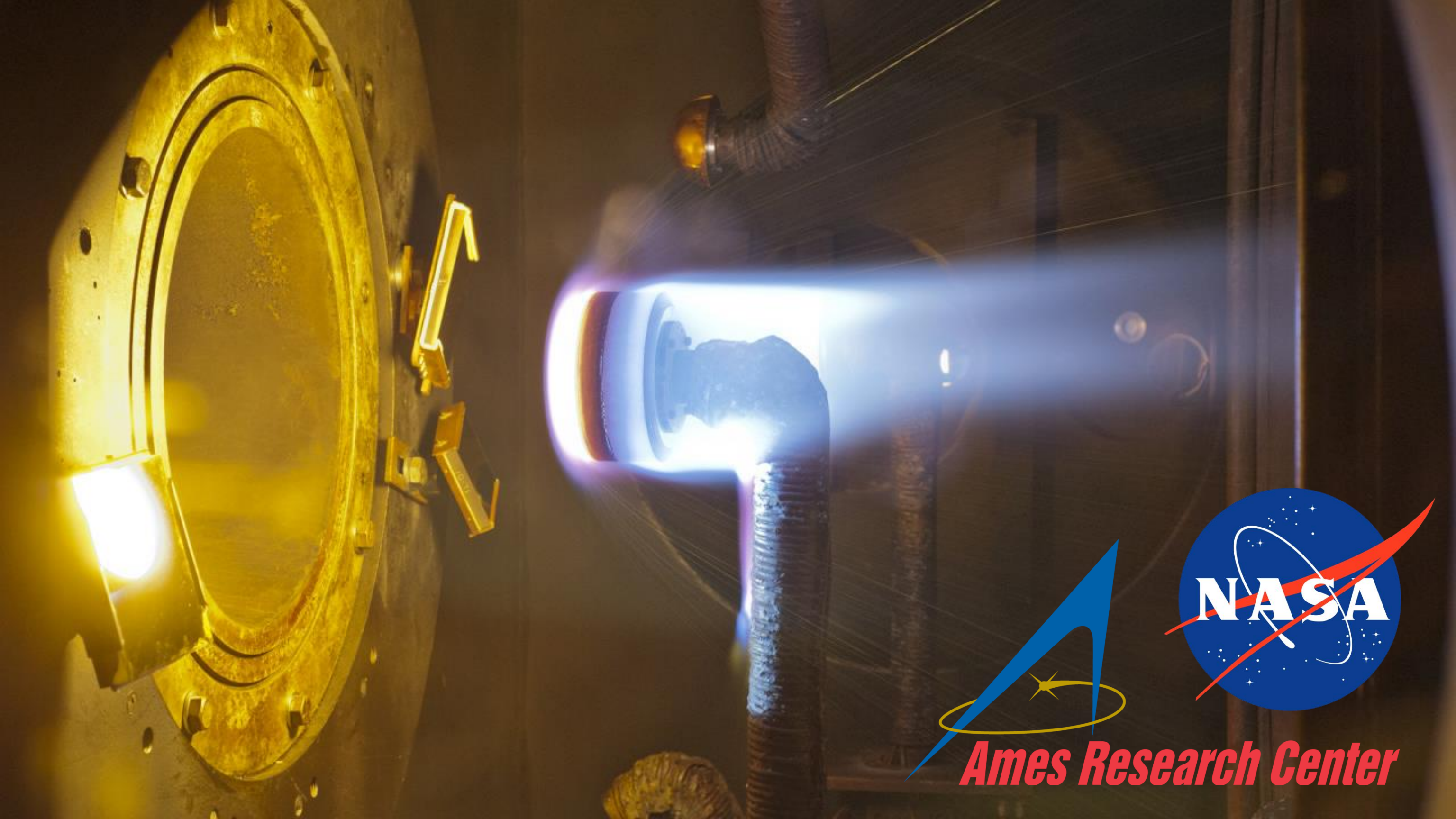
“NASA Grade” Software



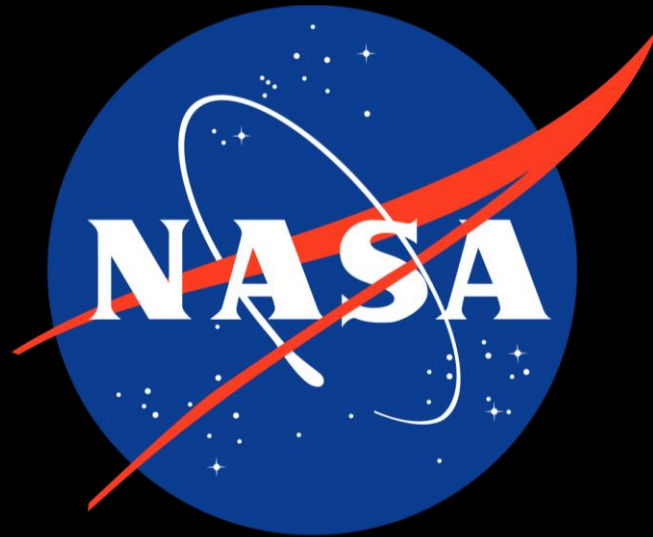


JKI









# What does “NASA Grade” mean?

\* *NASA Grade* is a metaphor, not yet another standard.



# Functional Requirements



# Functional Requirements

**Hardware**

**Process**

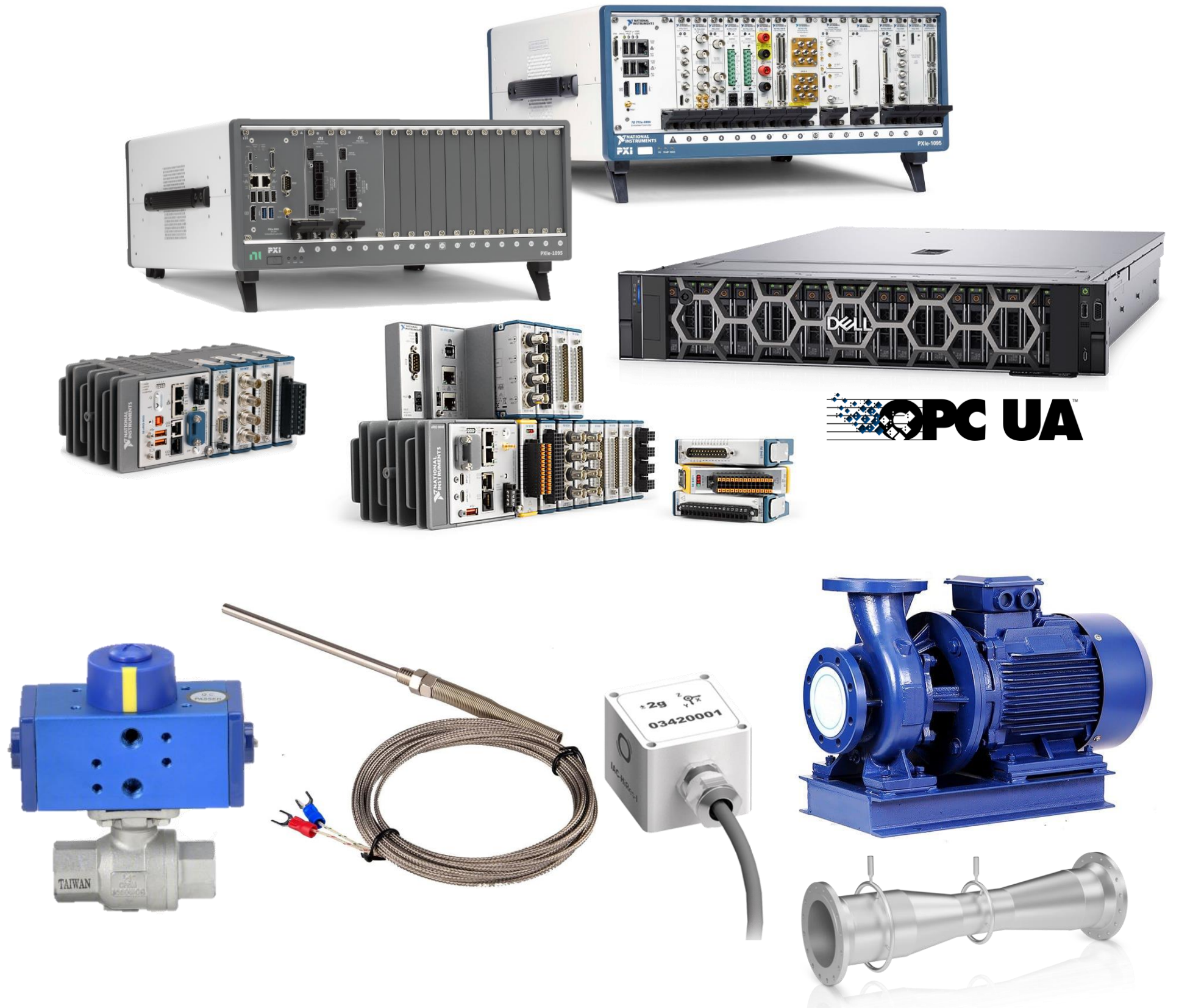
**Interface**



# Hardware

What are we connecting to?

- Channel Count
- Signal Types
- Data Rates
- Industrial Protocols
  - OPC, Modbus, Profinet
- Scaling
- Filtering

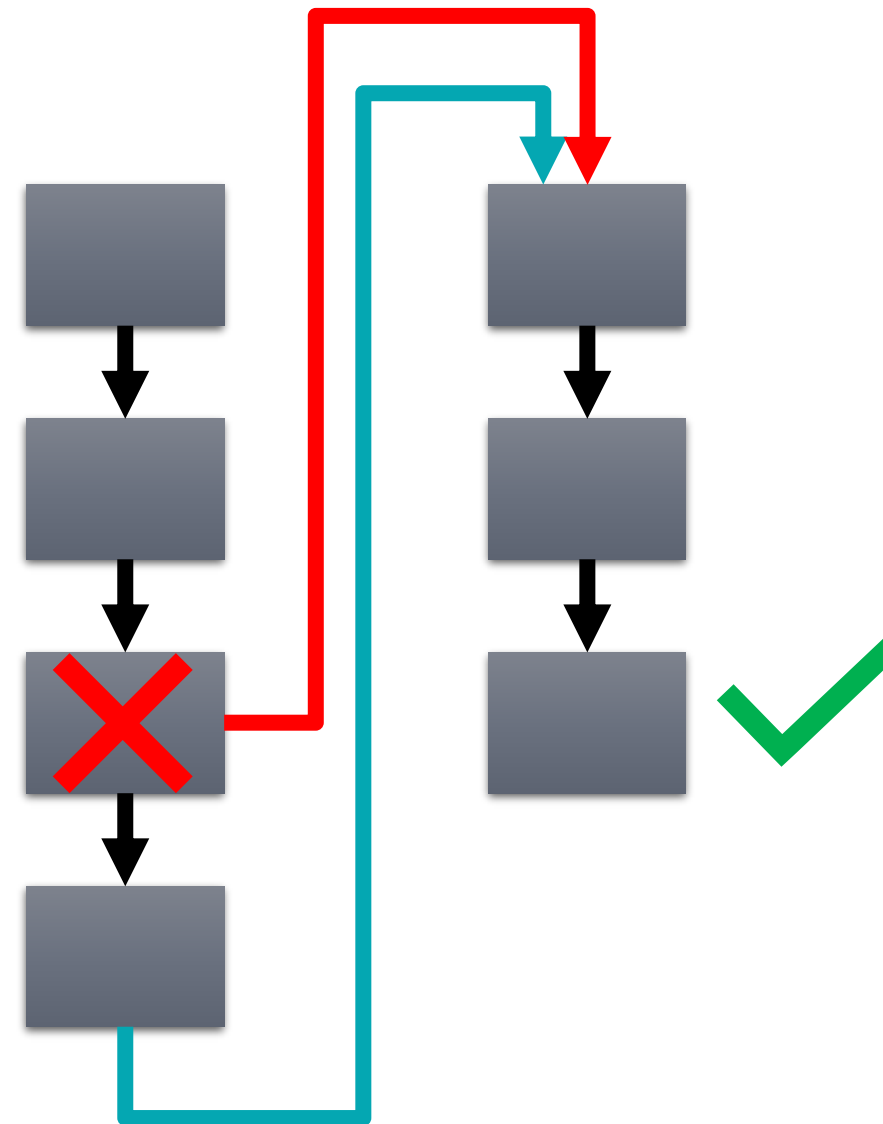


 **OPC UA**





- Step by step procedure
- Faults and Fallbacks
- User Input



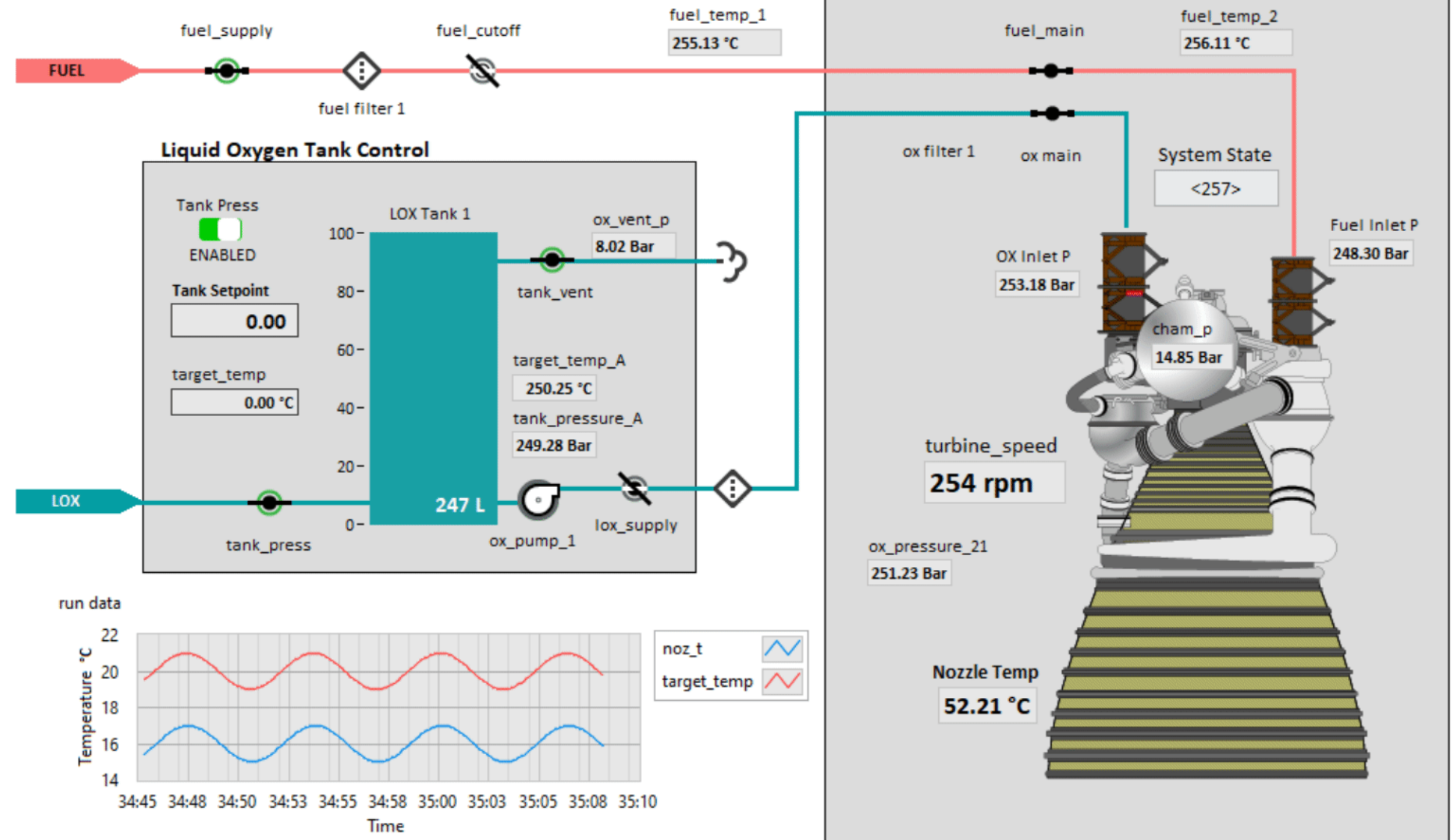


# Interface

## Application IO

- P&ID
- Channel Naming
- Manual Control
- Multi-display
- Data exports

### Propulsion Test Cell A

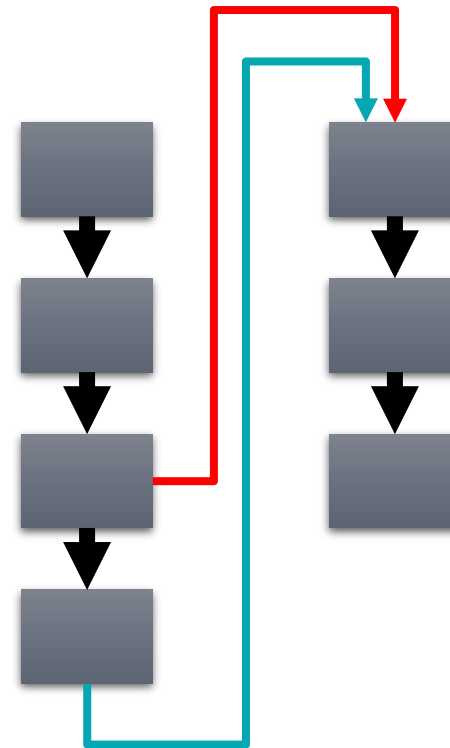




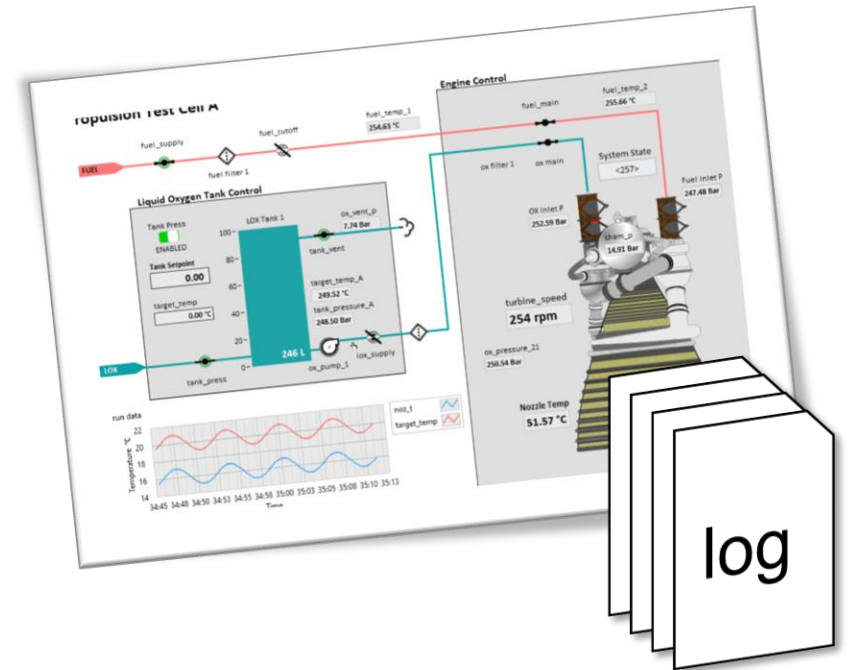
# Functional Requirements



Hardware

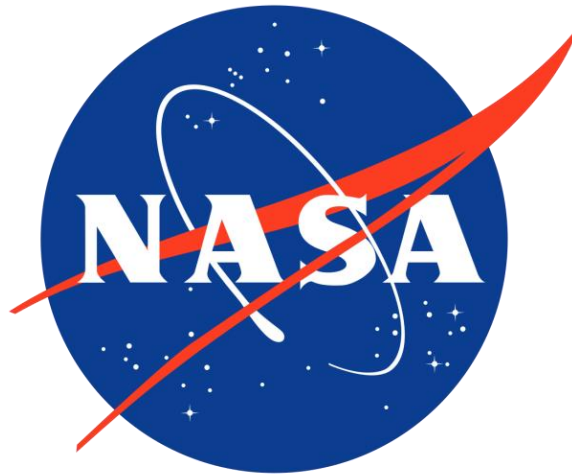


Process



Interface

# Who's Asking?

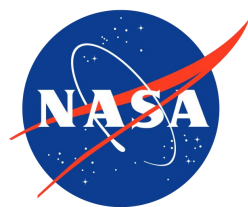




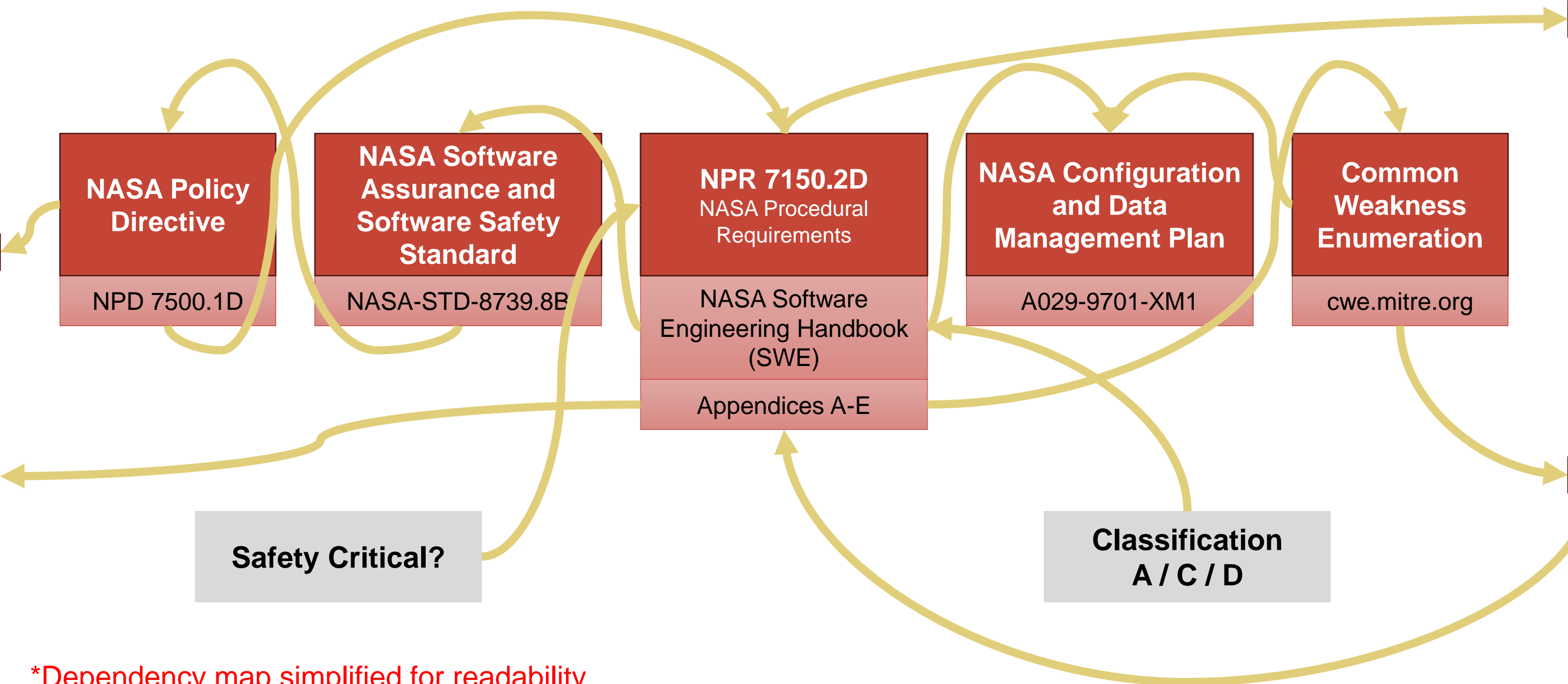
# Process Requirements

House rules





# NASA Requirements Documents



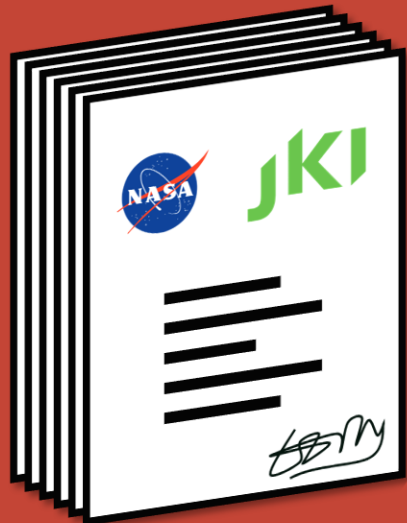
\*Dependency map simplified for readability



# SDP

## Software Development Plan

- 9000+ words
- 81 SWE Requirements
- 28 CWE Security Policies
- Signed & Tracked
- Binding



### Project Management

- Project Roles
- Developer Training
- Timelines
- Estimates
- Budgets
- Issue Tracking
- Issue Workflow
- Requirement Tracking
- Audits and Oversight
- Milestones

### Configuration Management

- Software Records
- Branches and Repos
- Source Code Control
- Environmental Configuration
- DevOps Infrastructure
- Deployment Configuration

### Software Design

- System Architecture
- COTs Software
- Software Reusability
- Sensitive Data
- Threat Assessment
- Software Classification

### Development Standards

- Performance Design
- Data Segregation
- Input Validation
- Error Handling
- Fail Safe Design
- Coding principals
  - LabVIEW Style
- Security Checklists
  - Low Level Memory Access
  - Network Communication
  - Encryption
  - File IO
  - System Commands
  - Authentication
- Unit Testing
  - Code Coverage
- Software Measurements
  - Performance Metrics
  - Static Analysis

### Verification and Validation

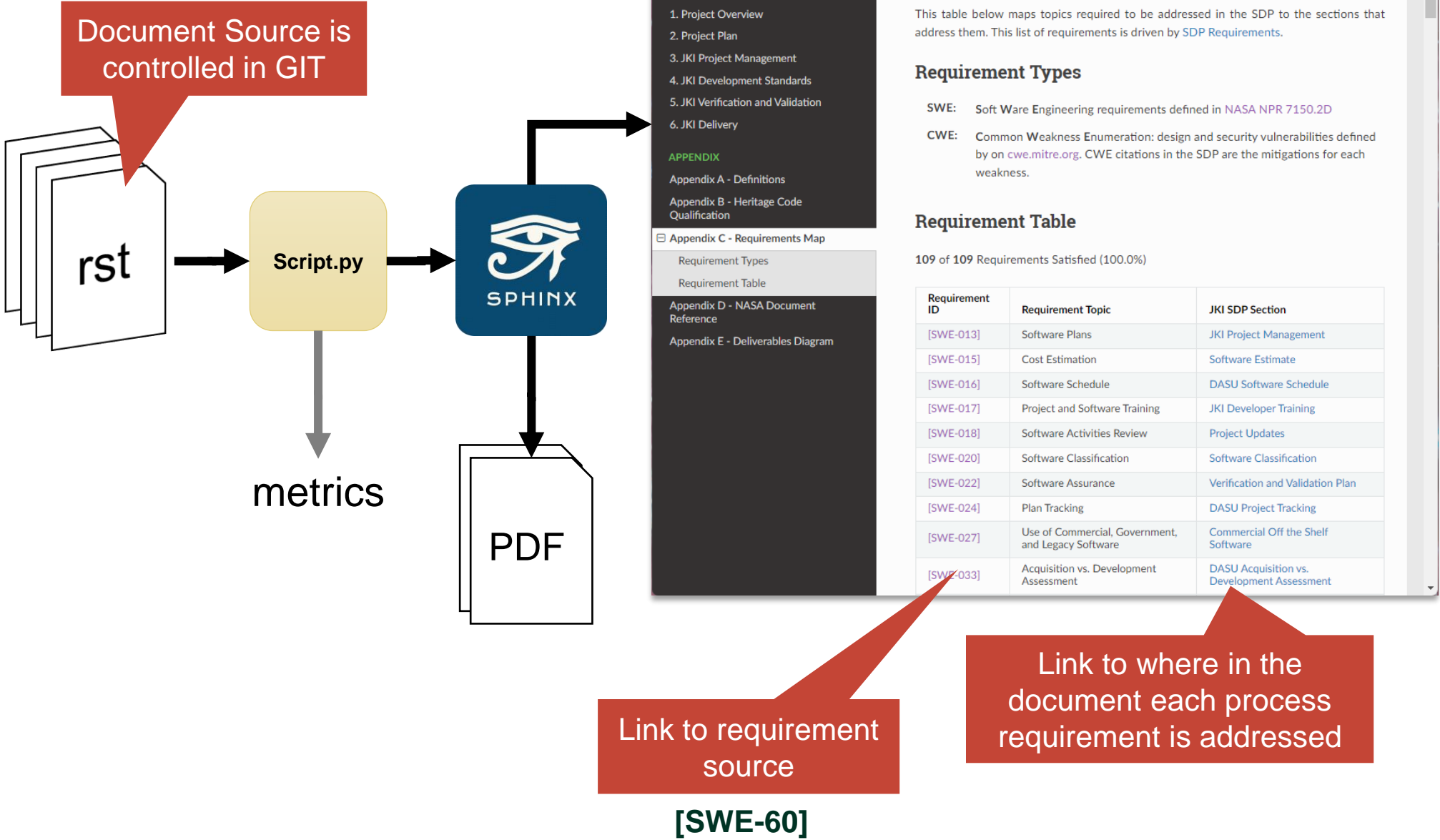
- Deliverables
- Peer Reviews
  - Pull Requests
  - Review Workflow
  - Review Checklists
- Integration Testing
- Delivery Verification
  - Verification Reports
- Hardware In The Loop Testing



# SDP

## The process behind the process

- Source Controlled
- Generated
- HTML & PDF
- Integrated Metrics





## Issue Tracking

- Status
- Priorities
- Roadmap
- Releases
- Users
  - Developers
  - Oversight



- ⚡ **Epics:** Overarching features
- ➕ **Features:** Things we need to add
- 🔴 **Defects:** Unexpected behavior

Find the Goldilocks effort level

# Requirements Tracking



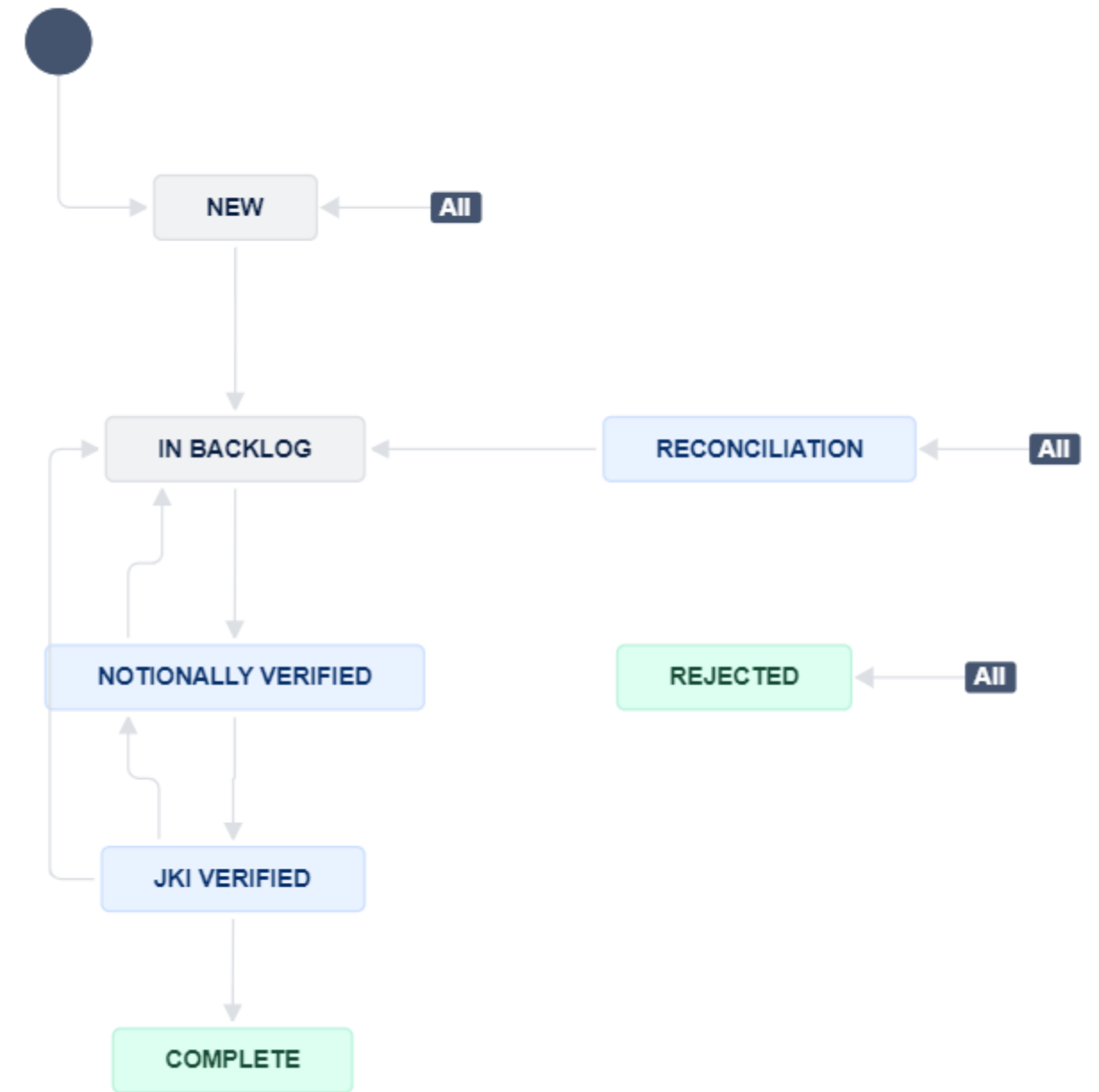
**Title:** The software shall...

- Capability
- Specification
- Interface
- Metric
- References 🔗

**Status:** In Work

**Verification Procedure:** TBD

**Bidirectional Traceability**





# Requirements Verification



## Inspection

- ✓ Verify encryption algorithm meets specification
- ✓ Verify 95% Uptime

## Unit Test

- ✓ Verify C to F converter, converts
- ✓ Verify a malformed config produces Error 5

## Integration Test (sim / HIL)

- ✓ Verify DAQ timestamps match
- ✓ Verify Shutdown command stops the app

## Manual Procedure

- 7.1.1 Close the **Tare Utility** Panel by *clicking* the **X** in the top right corner.  
**a. Verify** **Tare Utility** Panel closes  
☐Pass ☐Fail
- 7.1.2 In the **Channel Tree panel** *click* (**Slot 4 AI>s4 ai 01**)  
**a. Verify** the Tare scaler now displays the updated Offset value.  
☐Pass ☐Fail  
b. Record the **s4 ai 01** offset \_\_\_\_\_
- 7.1.3. **Select** the Popped out plot and view the data.  
**a. Verify** the graph now shows that the raw and scaled channels are offset by the Tare offset value. [\[REQ-237\]](#) [\[REQ-272\]](#)[\[REQ-236\]](#)  
☐Pass ☐Fail



## Functional Requirements



## Process Requirements




# **Assurance Requirements**



# Assurance Requirements

Coding Standards  
Code Review  
Unit Testing  
HIL Testing  
Static Analysis  
Performance  
Measurements  
Security Evaluation



# Coding Standards

- Quality
  - Code Smell
- Consistency
  - Code Review
  - Common tools
- Repeatability
  - Templates
  - Style Guide

Directive	NEVER Break
Principal	Only break if you have a good, documented reason
Guideline	Do it when you can
Tip	Preferences and syntactic sugar





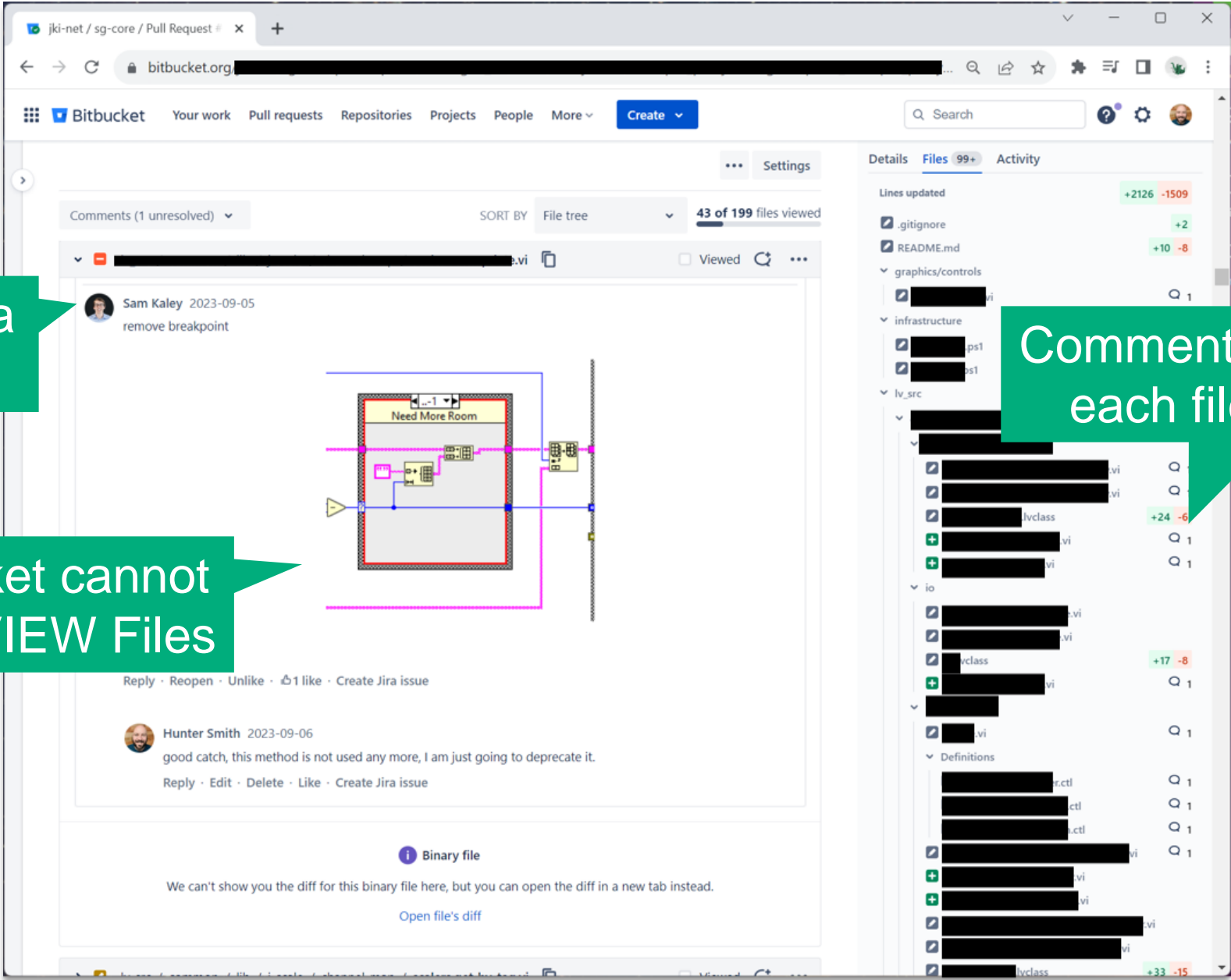
# Code Review

- Developer Standards
- Coding Style
- Error Handling
- Bugs
- Performance
- Readability
- Documentation

Caught a bug

! Bitbucket cannot diff LabVIEW Files

Comment on each file

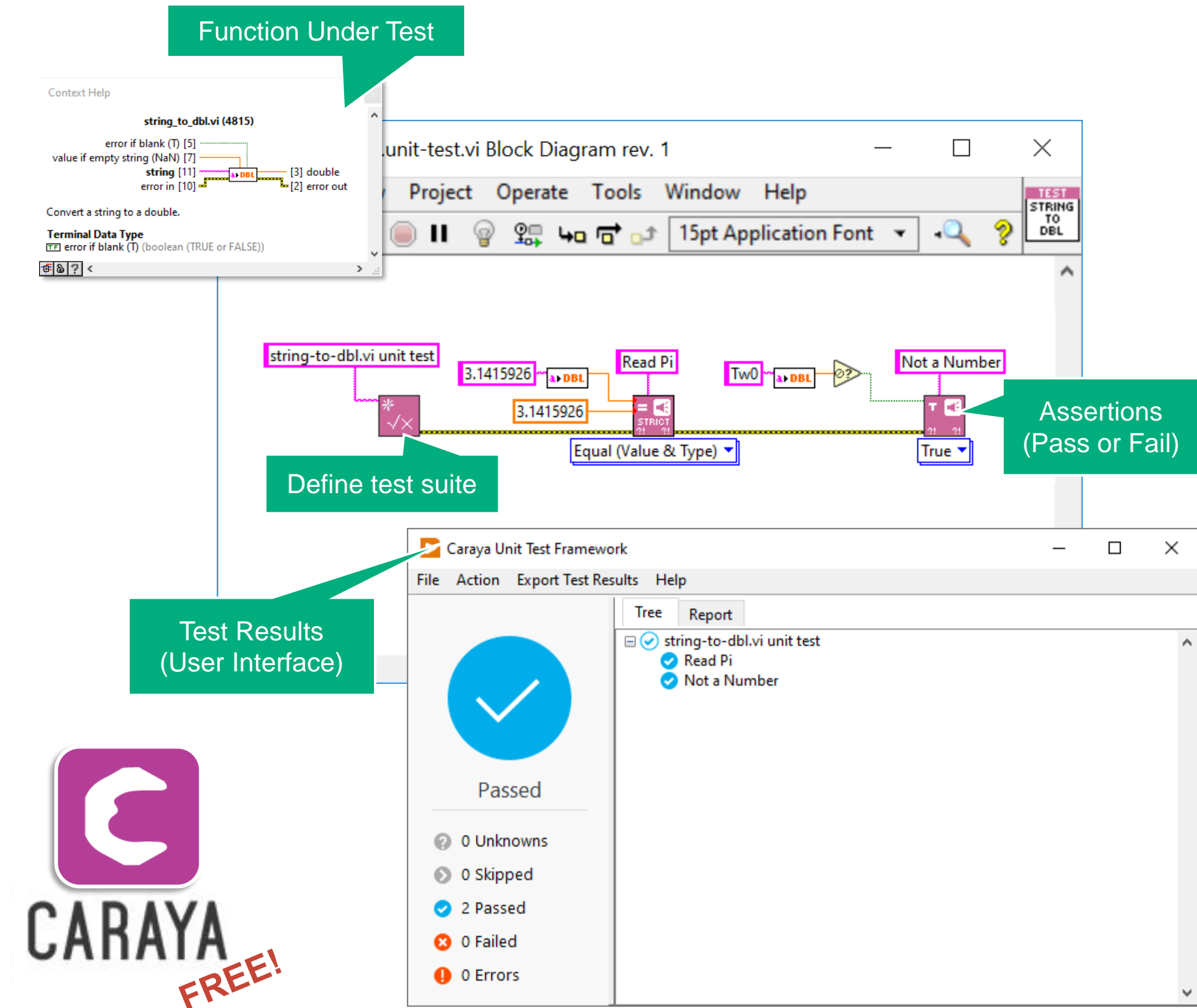




# Unit Testing

## Part 1

- Define the expected behavior
- Define, Assert
- Guarantee the interface





# Unit Testing

## Part 2

- Verify Faults
- Repeatable tests
- Easy to expand
- Define the corners

### Test Passing and Failing cases

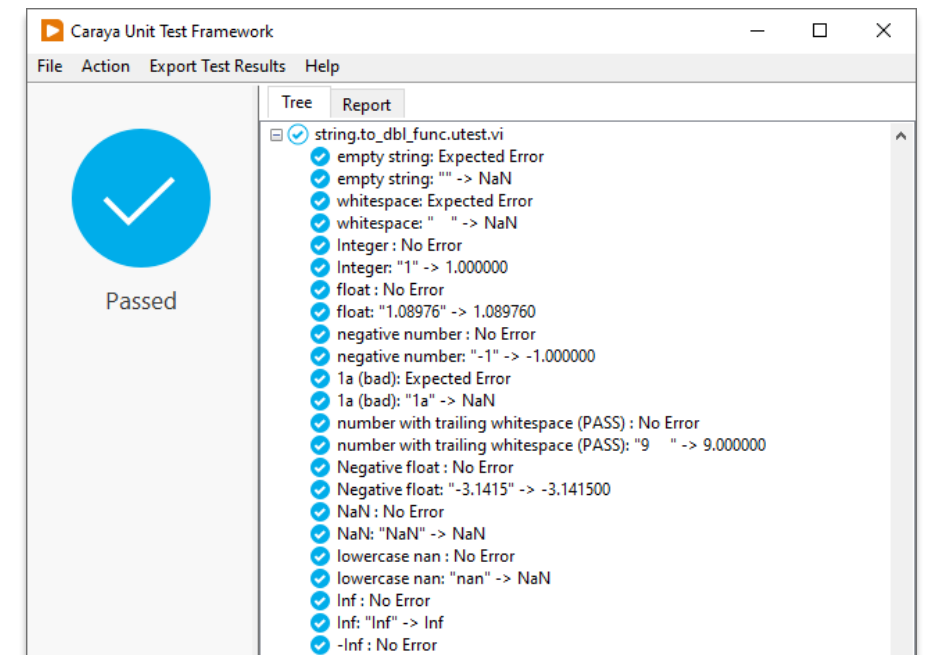
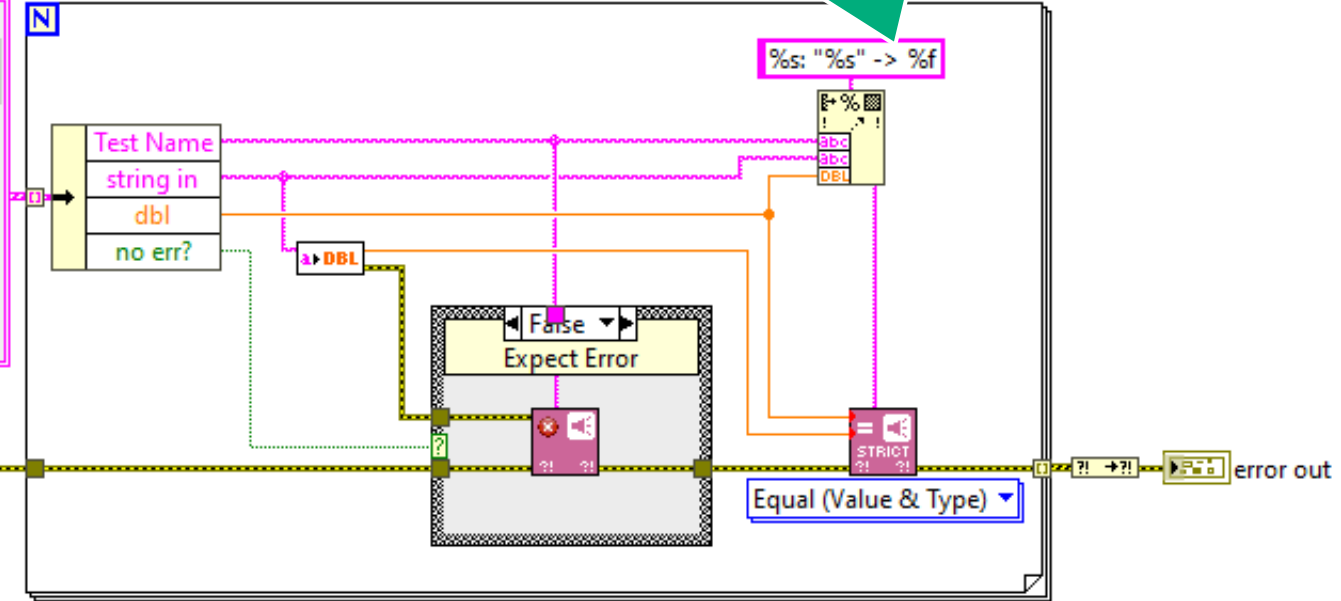
Test Cases

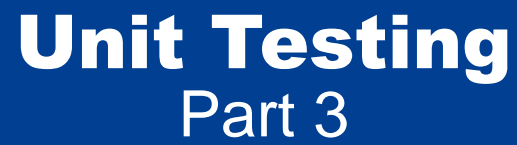
Test Name	string in	no err?	dbl
empty string		F	NaN
whitespace		F	NaN
Integer	1	T	1
float	1.08976	T	1.08976

Easy to add test cases

### Dynamic Tests Names

For Each Test Case



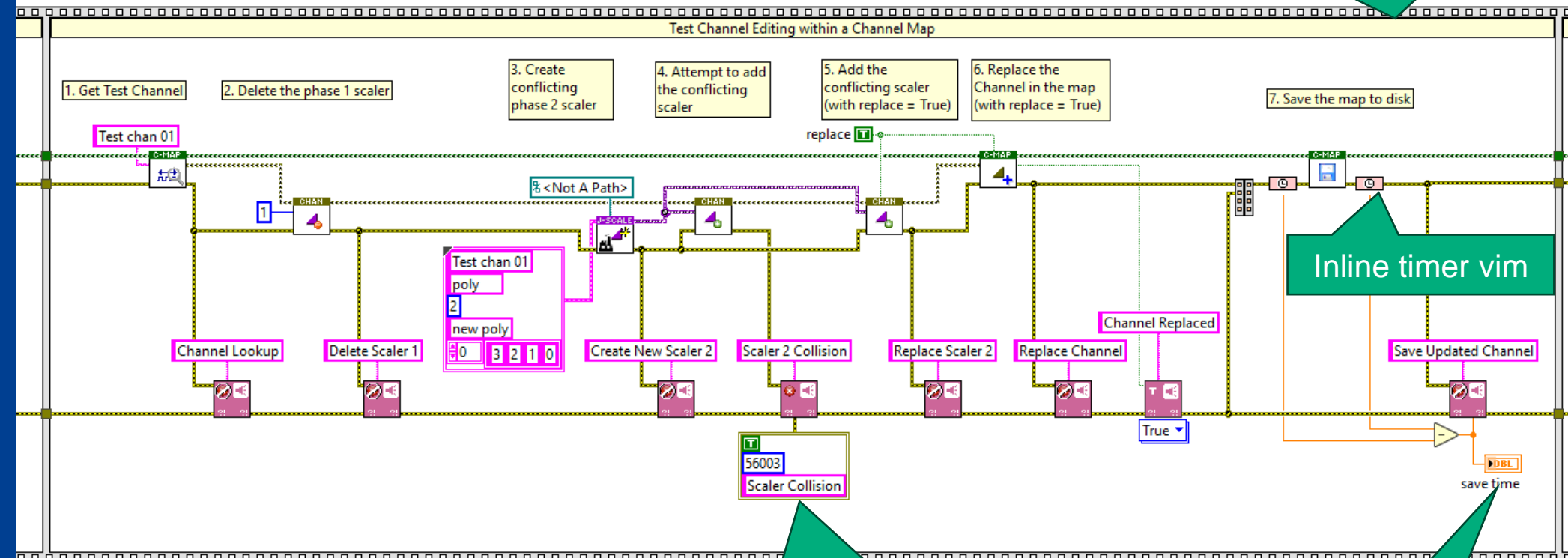


- Subsystem testing
- Sequence a runtime behavior step by step
- Verify nominal behavior and faults

## Inline timer vim

## Verify code generates a specific error

## Performance Metrics

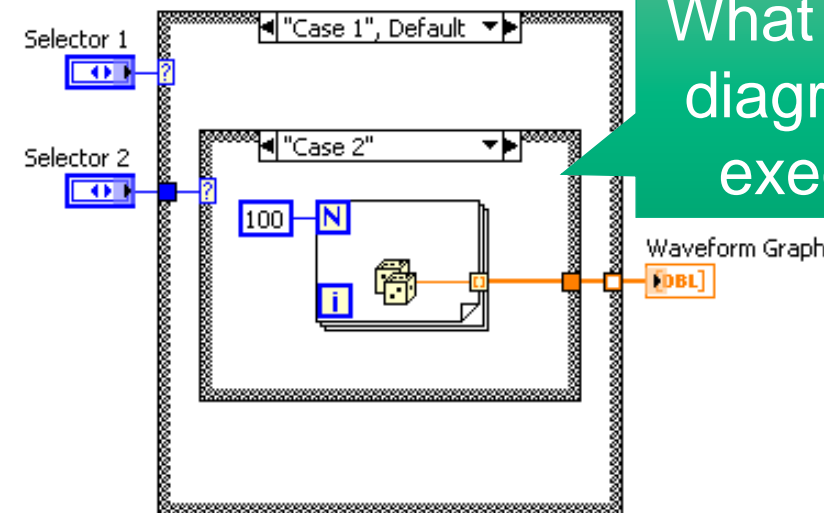
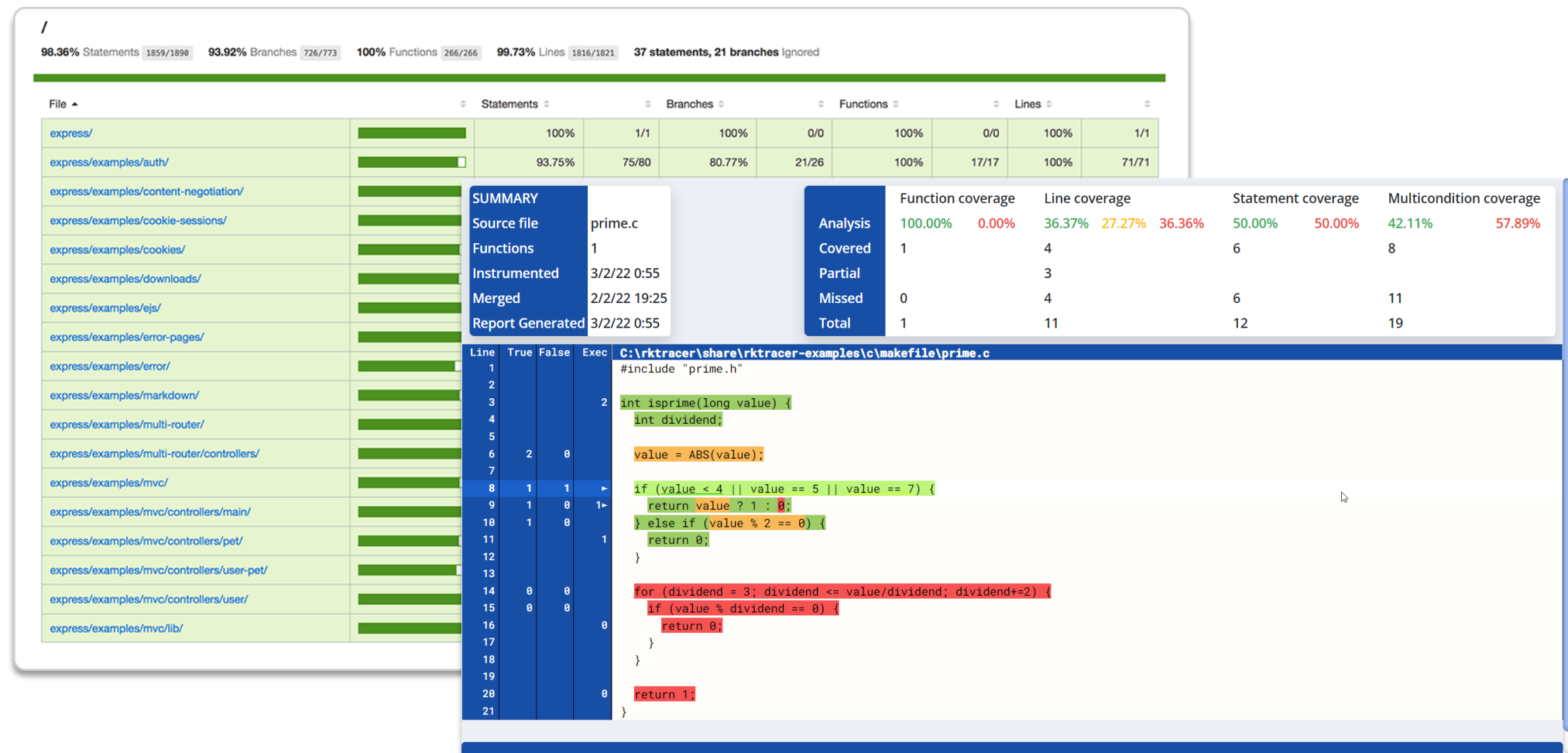




# Code Coverage

measure your tests

- How much code are you testing?



What % of the diagram was executed?





# J-Tester

Caraya +

- Caraya runner
- Code Coverage
- Artifact Handling



## Bitbucket

Download Reports  
Test Results+ Code Coverage

JKI / ... / Pipelines

✓ #121 Rerun

2f82fe5 [REDACTED]

main

[Learn more about reports](#)

13min 46 sec 3 days ago

Pipeline

✓ Caraya Unit Tests  
6m 54s • 1774 tests passed • [↗](#)

✓ J-Crawler Static Analysis  
1m 30s • [↗](#)

Build Artifacts

=====

jki.net

=====

j-tester-Caraya

=====

Discovering Tests  
110 Caraya Test VIs discovered (96.3879s)  
30 total test VIs ignored due to regex

Starting Tests!  
Testing Complete 278.526s

-----

Test Results

-----

372 Tests  
1774 Test Assertions  
ALL 1774 PASSED  
Test Duration: 278.526s  
Test Report: test-results.xml

-----

Code Coverage Summary

-----

779 /2856 VIs measured  
75 VI's exempt from code coverage.  
Mean coverage: 24.85%  
Coverage Report: coverage-report.csv

Know what  
**IS** and **IS NOT**  
being tested

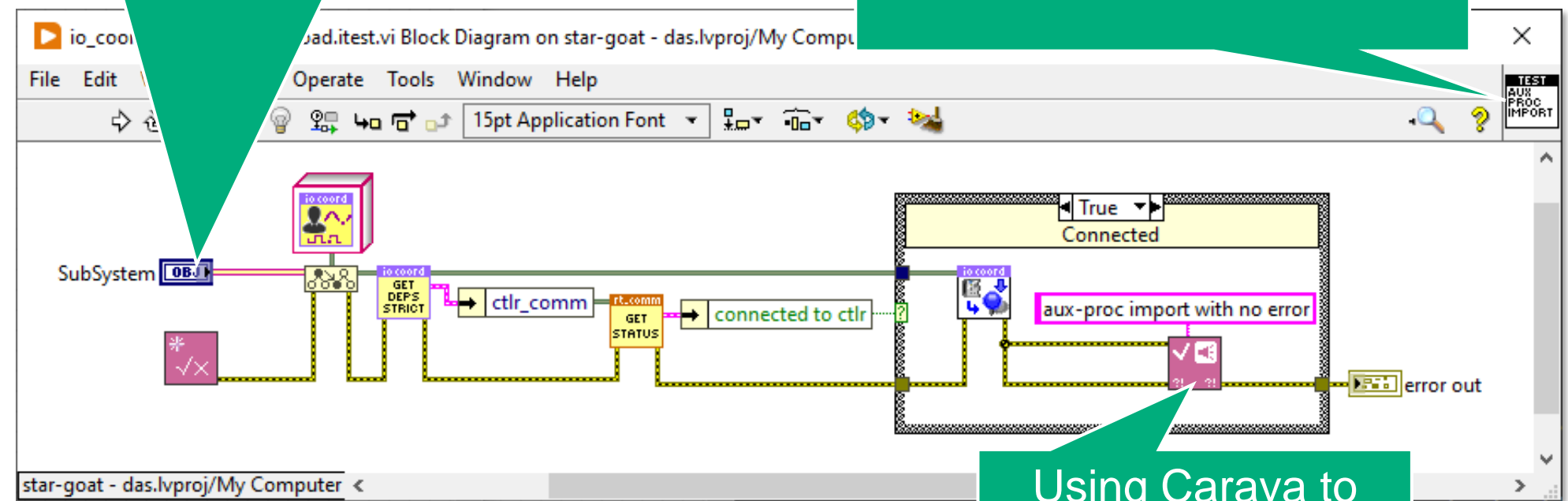


# Integration Testing together now

- Test whole subsystems
  - data paths
  - error handling
  - Interfaces
- T.I.E Test Modes
  - Client
  - Controller
  - Client + Controller (dev)
  - Client (dev) + Exe

Requires a Framework to Instantiate the Object Under Test

Framework spins up System then runs each tests



Using Caraya to Capture Results

STAR-GOAT Test Integration Engine

Test Type: sg-combined

Test Definitions:

- ctrl\_comm\_das.itest.ini
- das-command-hud\_das.itest.ini
- io\_coordinator.das\_com-check.itest.ini

Test Cases ("": all)

Running...

STAR-GOAT Test Integration Engine (T.I.E.)

Arguments:

- Name: Default Test
- Test Type: sg-combined
- Test Names: (run all)
- Coverage Report Path: C:\Users\hunter\Documents
- Test Report Path: C:\Users\hunter\Documents\
- Ext exe name:
- No Pre-Load VIs
- Repo Root: C:\code\sg-das

Tests Loaded:

- 1 system groups
- 1 total test groups
- 3 total test cases
- ctrl\_comm.das\_status\_read.itest.vi
- das-command-hud.das\_capture-run.itest.vi
- io\_coordinator.das\_com-check.itest.vi

Running Tests...

Loading System Definition 1:

Caraya Unit Test Framework

Tree Report

- TIE
- TIE.interface.lvclass:setup
- TIE.sg-client-ctrl.lvclass:create
  - Ctrl - Init
    - controller initialized: No Error
    - network comms established
    - take cmd: No Error
    - ctrl.error initialize: No Error
- das-default.sg-client.xml
  - Client - Init
    - Wait for Connection: No Error
    - connected to controller
    - Take Control: No Error
    - ctrl\_system.command.status.read: No Error
    - Assert Equal Value\_Variant
    - ctrl\_system.command.status.read: No Error
    - ctrl Version: 20.0.1f1
    - ctrl.error initialize: No Error
- das-demo1.sg-manifest.ini
  - TIE.sg-client.lvclass:ctrl\_manifest.load.vi
    - manifest: export\_to\_controller
    - manifest: import\_from\_controller
    - manifest: hash match
- Client - Start
  - ctrl.start: No Error

0 Unknowns

0 Skipped

15 Passed

1 Failed

0 Errors



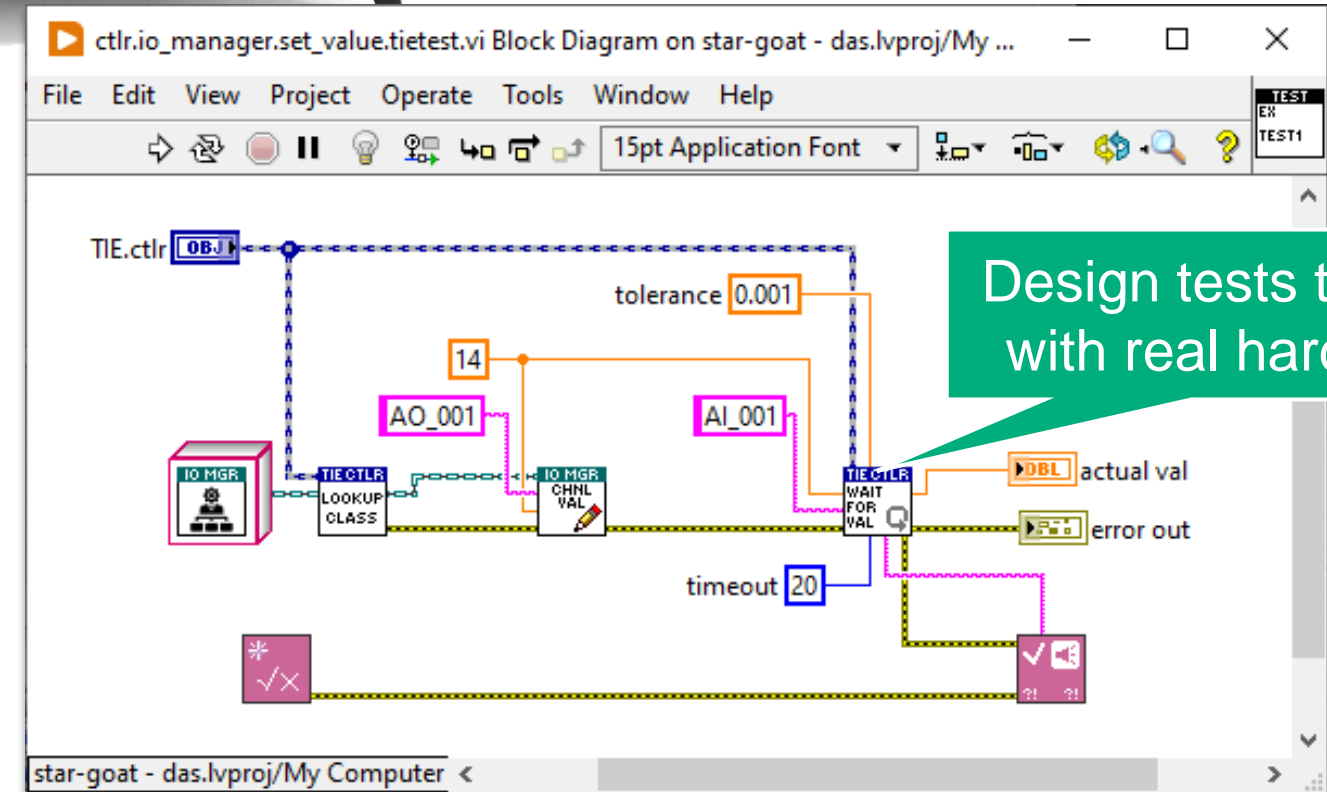
# HIL Testing

Real IO

- Run on window CI targets
- Deploy to RT targets
- Loopback testing
- External Testing
- Dogfooding

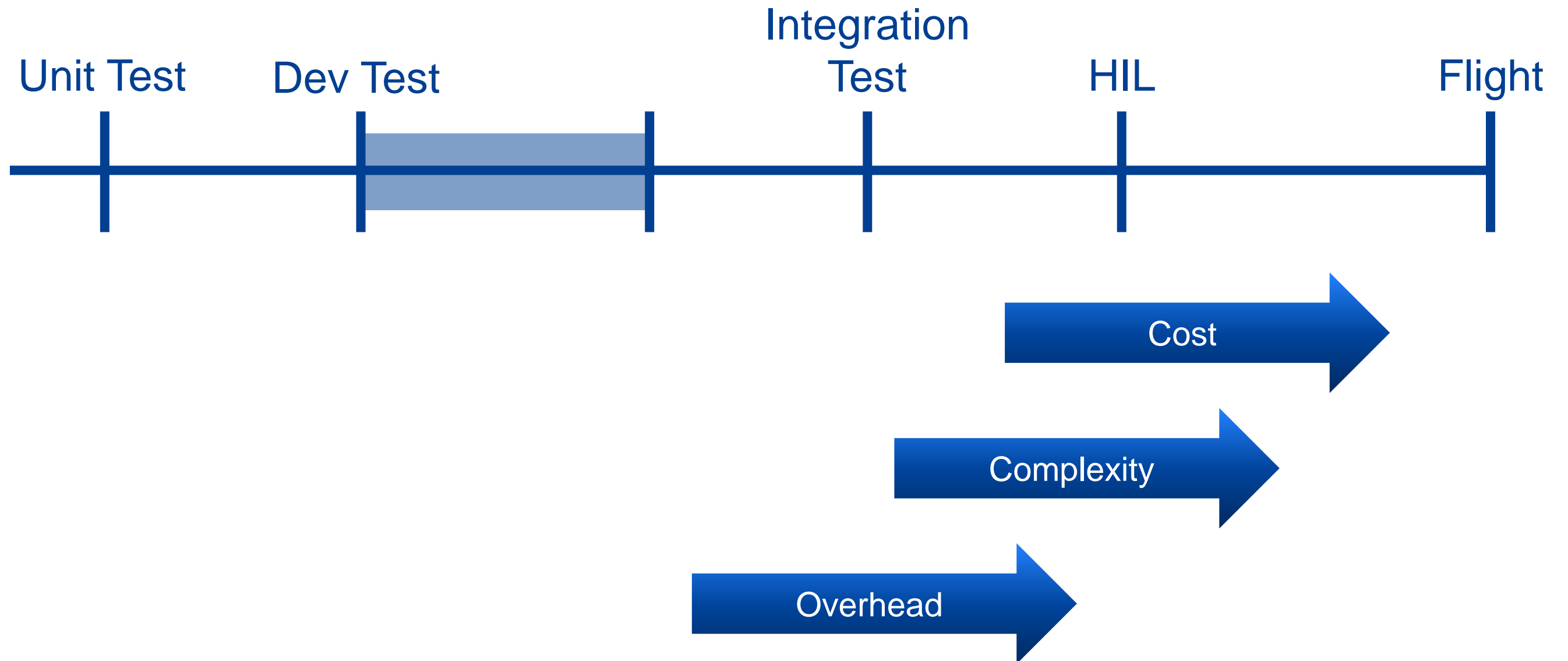


Run Tests on hardware



Design tests to work with real hardware

# Test Like You Fly



# Static Analysis

Never miss a broken VI

- Catch Mistakes
- Enforce Standards
- Audit Style
- Automatic Fixes
- Static Analysis

## Automate Quality!

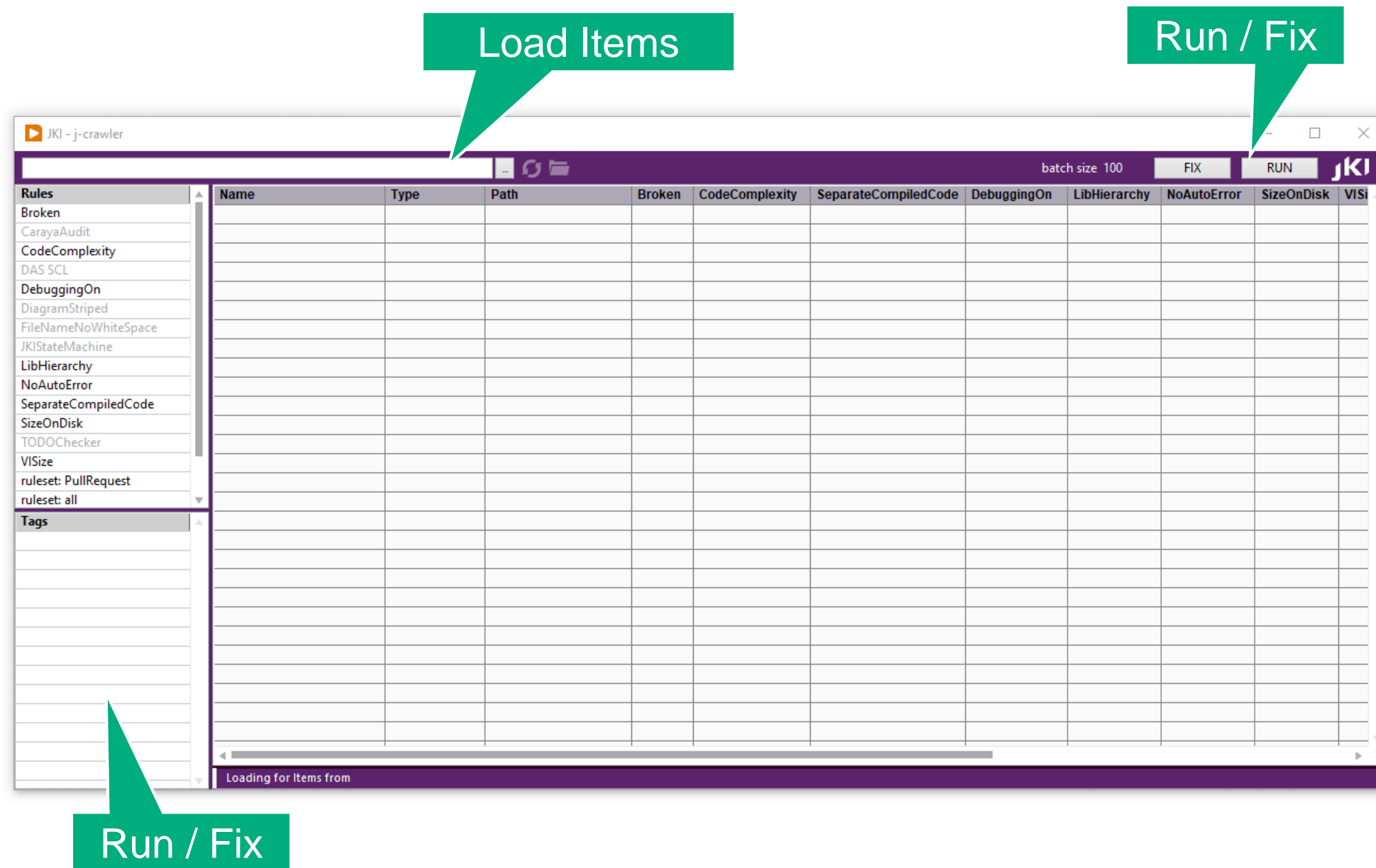




# J-Crawler

Static Analysis Utility

- Load dir, class, Ivproj
- Run “Rules”
- Audit
- Fix!
- Beta Available Soon





# J-Crawler

## CI Static Analysis

- Check on every commit
- Require pass to merge
- Easy to read results
- Easy configuration
  - Allow list
  - Rulesets
- Helpful Artifacts



Bitbucket

JKI / ... / Pipelines

✓ #121 Rerun

2f82fe5 main

[Learn more about reports](#)

13min 46 sec 3 days ago

Pipeline

Caraya Unit Tests  
5m 25s • 1744 tests passed • [↗](#)

J-Crawler Static Analysis  
8m 15s • [↗](#)

Build Artifacts

Discovering Items...

Discovered 5,693 Items: (36.9s)

5,693 Total Files  
4,220 LabVIEW VIs  
831 LabVIEW Controls  
228 LabVIEW Classes  
3 Markdown Files  
30 CSV Files  
33 ini Files  
27 XML Files  
9 TDMS Files  
3 VIPC Files

CRAWLING...

CRAWL Complete 455s

j-crawler Rule Results

22,774 Total Results

Results by State

21,845 Passed Results  
164 Skipped Results  
16 Failed Results  
748 Info Results  
1 Errored Results

Results by Rule

(Times are unfolded parallel execution so may exceed run time)

Broken : 4910 Passed, 141 Skipped, 6 Failed (25.8s)  
DebuggingOn : 4216 Passed, 2 Failed (18.5s)  
LibHierarchy : 3463 Passed, 7 Failed, 748 Info (21.0s)  
NoAutoError : 4214 Passed, 4 Skipped (20.3s)  
SeparateCompiledCode : 5042 Passed, 19 Skipped (19.5s)

FAILURES

16 Failed Results

lv\_src\client\client-common\lib\lookup.vi :: LibHierarchy :: Mispalaced  
lv\_src\common\lib\j-console\filter\_verb.vi :: LibHierarchy :: Mispalaced  
lv\_src\common\lib\j-scale\agent\scale.vi :: DebuggingOn :: (Disabled)  
lv\_src\common\lib\j-scale\channel\scale.vi :: DebuggingOn :: (Disabled)  
lv\_src\common\lib\name.validate.vi :: SeparateCompiledCode :: (Not Seperated)

Download Reports

What's in this Repo?

Require Passing Tests to Merge

See the Error in your Email

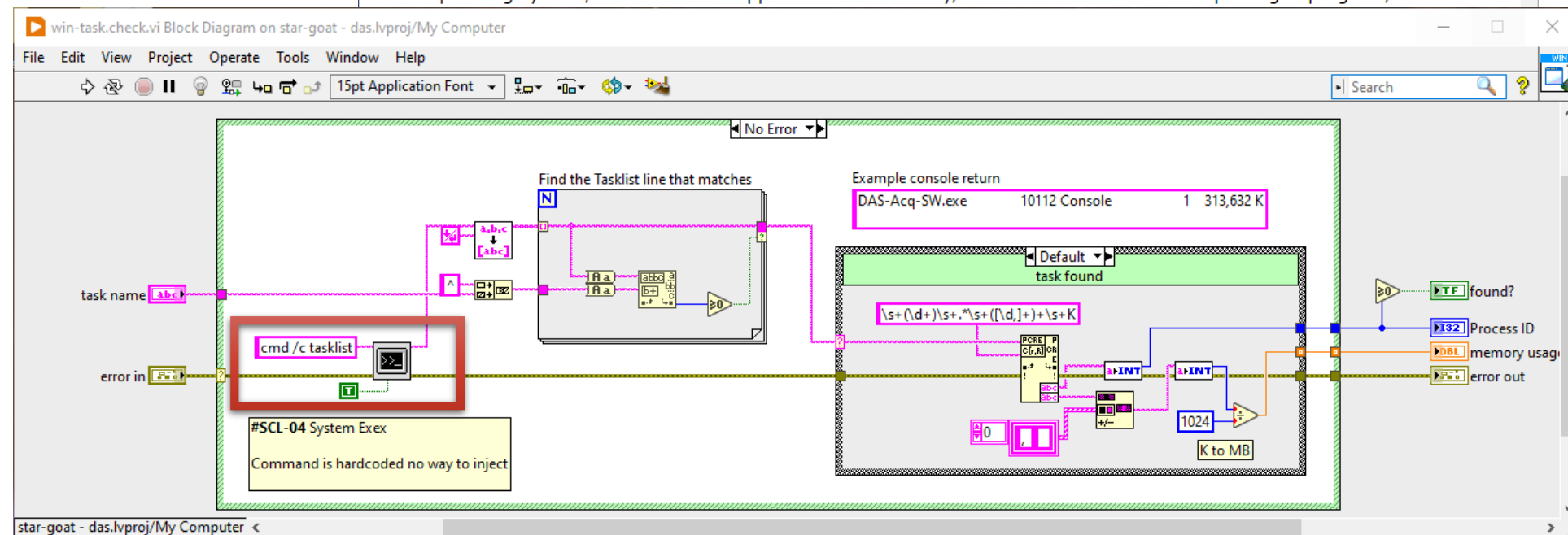


# Security

## Audit with J-Crawler

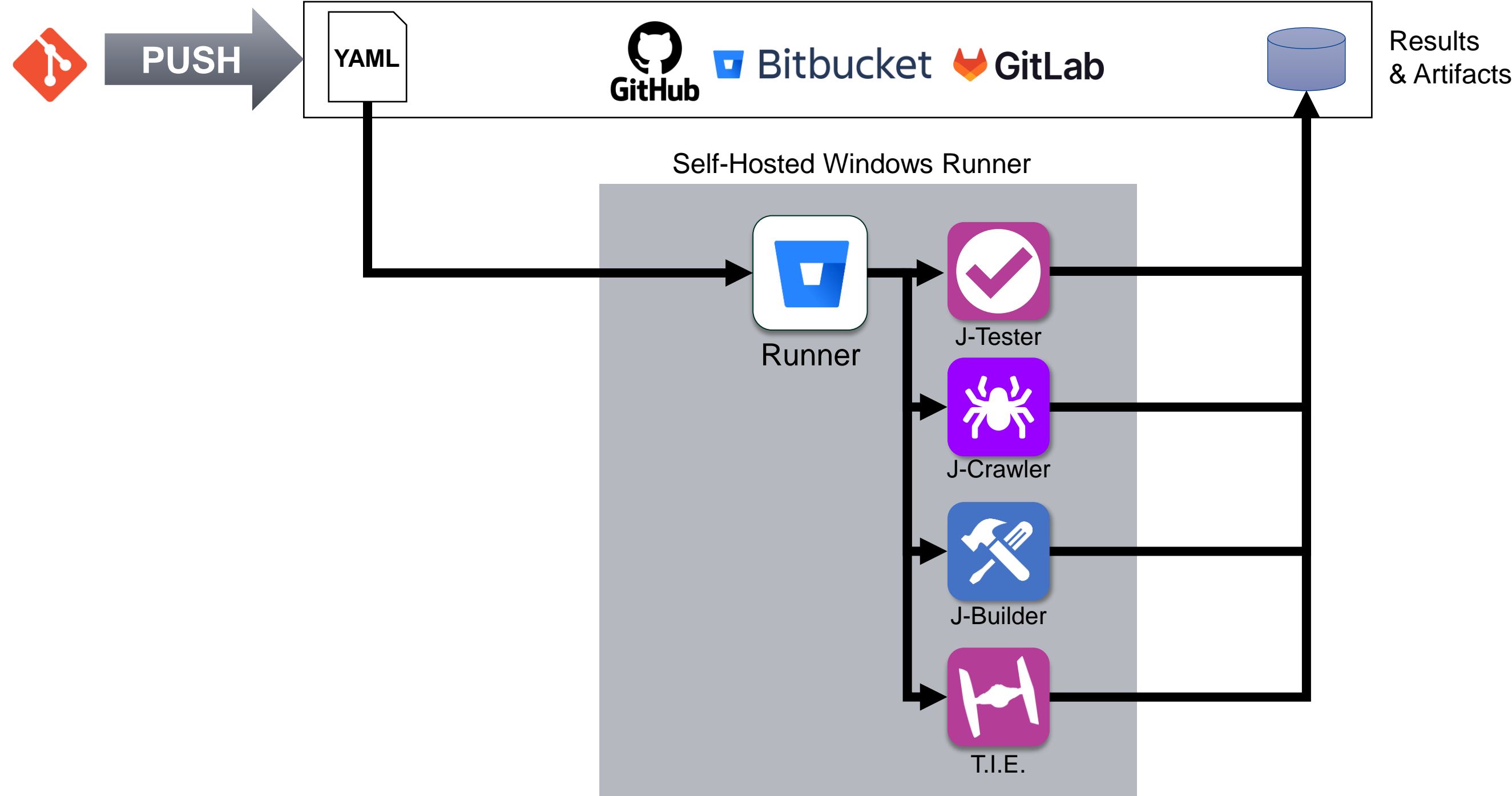
- Identify every use
- Security Checklist
  - File IO
  - Network Communication
  - Authentication
  - OS Commands
  - Memory

The screenshot shows the MITRE CWE website page for CWE-78. The page title is "CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')". It includes a "Weakness ID: 78" and a "Vulnerability Mapping: ALLOWED" status. The "Description" section states: "The product constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component." The "Extended Description" section states: "This could allow attackers to execute unexpected, dangerous commands directly on the operating system. This weakness can lead to a vulnerability in environments in which the attacker does not have direct access to the operating system, such as in web applications. Alternately, if the weakness occurs in a privileged program, it..."



Under Development

# Automate with CI/CD





#1740

Rerun

🔗 [ebe99a2](#) Merged in [bugfix/JDEV-283-threshold-alert-performance](#) (pull request #91) Bugfix/...

🔗 [sdp](#)

custom: All

🔗 [Bugfix/JDEV-283 threshold alert performance](#)

📖 [Learn more about reports](#)

🕒 1hr 8min 0 sec 📅 19 hours ago



## Pipeline



✓ J-Crawler Static Analysis  
12m 58s • 🔗

✓ Caraya Unit Tests  
10m 56s • 3319 tests passed • 🔗

✓ T.I.E - Controller  
7m 11s • 32 tests passed • 🔗

✓ T.I.E - Client  
8m 17s • 10 tests passed • 🔗

✓ T.I.E - Combined  
8m 33s • 38 tests passed • 🔗

✓ Build Star Goat Client  
17m 45s • 🔗

✓ Build Star Goat Controller  
10m 30s • 🔗

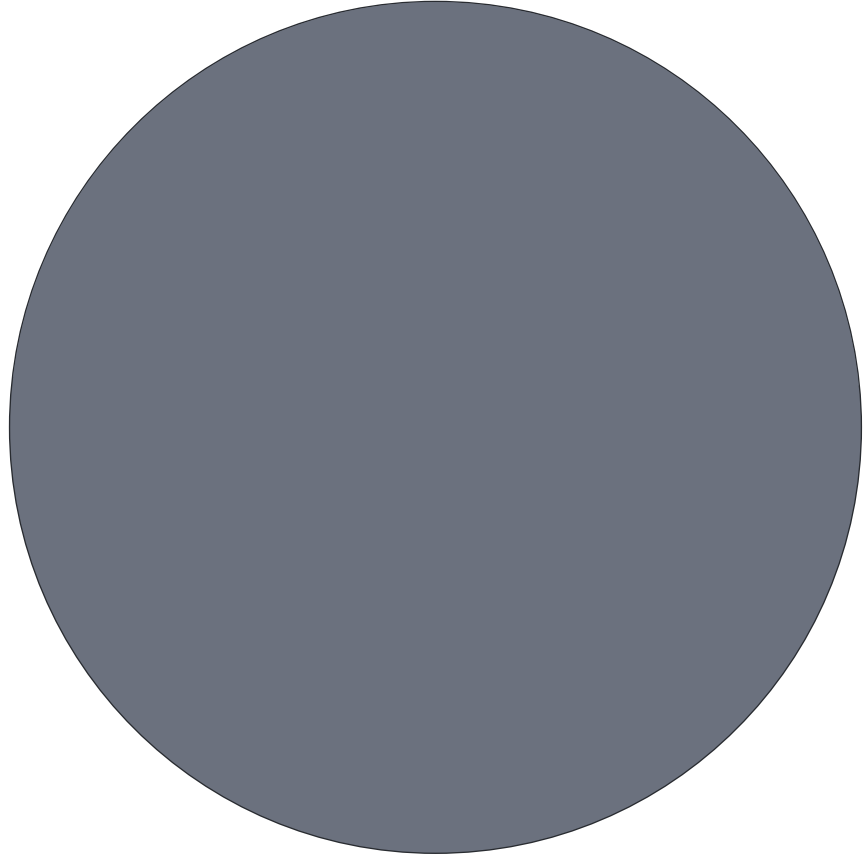
✓ Build Process Flow VIPC  
1m 45s • 🔗

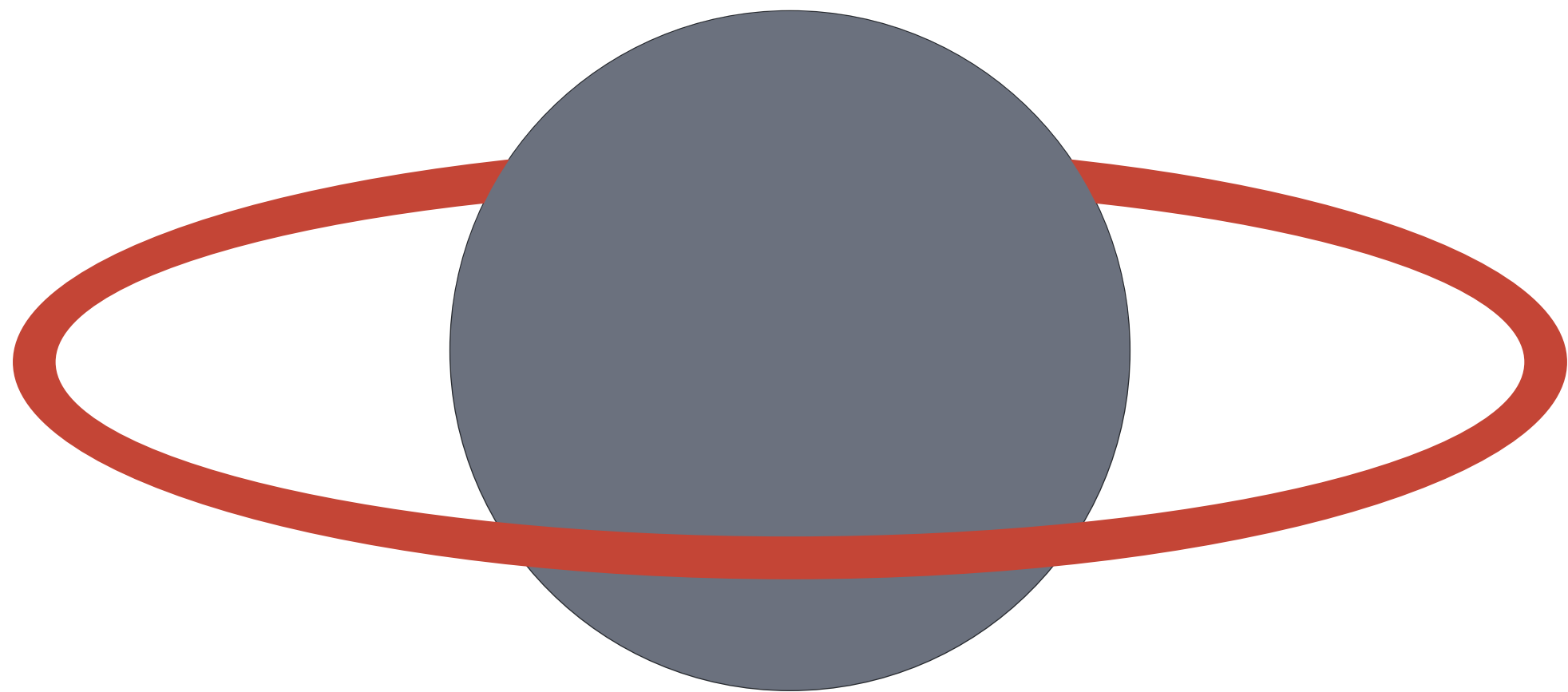
## Build Artifacts

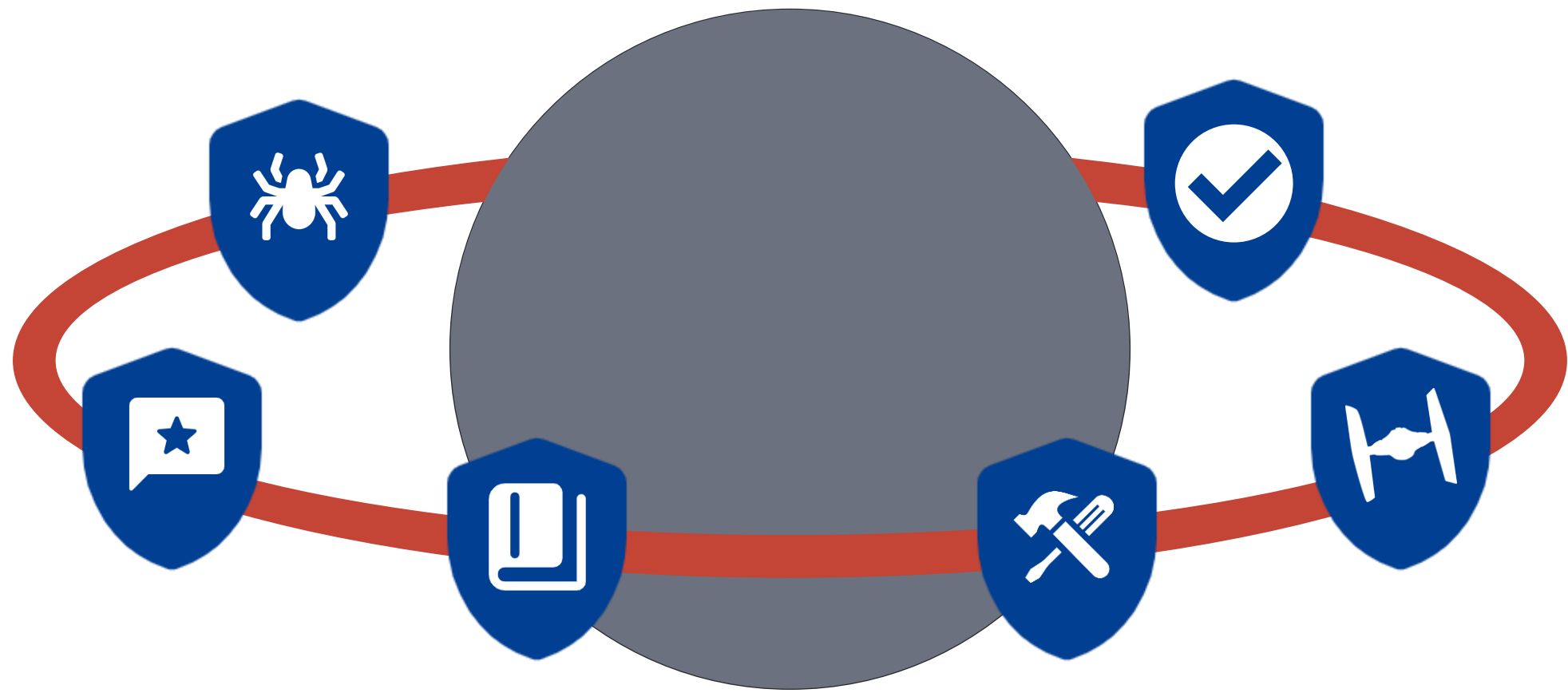


```

11  =====
12
13  _\  _\  _\
14  _\  _\  _\  _\
15  _\  _\  _\  _\
16  _\  _\  _\  _\
17
18  =====
19  STAR-GOAT
20  Test Integration Engine
21  (T.I.E.)
22  =====
23
24  Arguments
25      Name: TIE-Controller-Bitbucket
26      Test Type: sg-ctrlr
27      Test Names: (run all)
28      Coverage Report Path: C:\rnr\pipe\temp\09b9bba7-8bc9-50e8-a537-
29      81057cfd3f98\1715894834878\build\test-reports\tie-ctrlr-coverage-report.csv
30      Test Report Path: C:\rnr\pipe\temp\09b9bba7-8bc9-50e8-a537-
31      81057cfd3f98\1715894834878\build\test-reports\test-report.xml
32      Ext exe name:
33      No Pre-Load VIs
34      Repo Root: C:\rnr\pipe\temp\09b9bba7-8bc9-50e8-a537-
35      81057cfd3f98\1715894834878\build
36
37  Tests Loaded
38      2 system groups
39      3 total test groups
40      6 total test cases
41          aux-proc.nominal_launch.tietest.vi
42          ctrlr.io.opc_ua.close_opc.tietest.vi
43          ctrlr.io.opc_ua.open_opc.tietest.vi
44          ctrlr.io.opc_ua.set_value.tietest.vi
45          ctrlr.io_manager.get_io_list.tietest.vi
46          ctrlr.io_manager.set_value.tietest.vi
47
48  -----
49  Running Tests...
50
51  Loading System Definition 1:
52      aux-proc.sg-client.xml : 1 Test Groups
53      Loading Ctrlr Manifest: TIE-aux-proc.sg-manifest.ini
54      Groups Complete (8.128s)
55  Loading System Definition 2:
56      default.sg-client.xml : 2 Test Groups
57      Loading Ctrlr Manifest: TIE-configOPC-UA.sg-manifest.
58      Loading Ctrlr Manifest: TIE-demo1.sg-manifest.ini
  
```





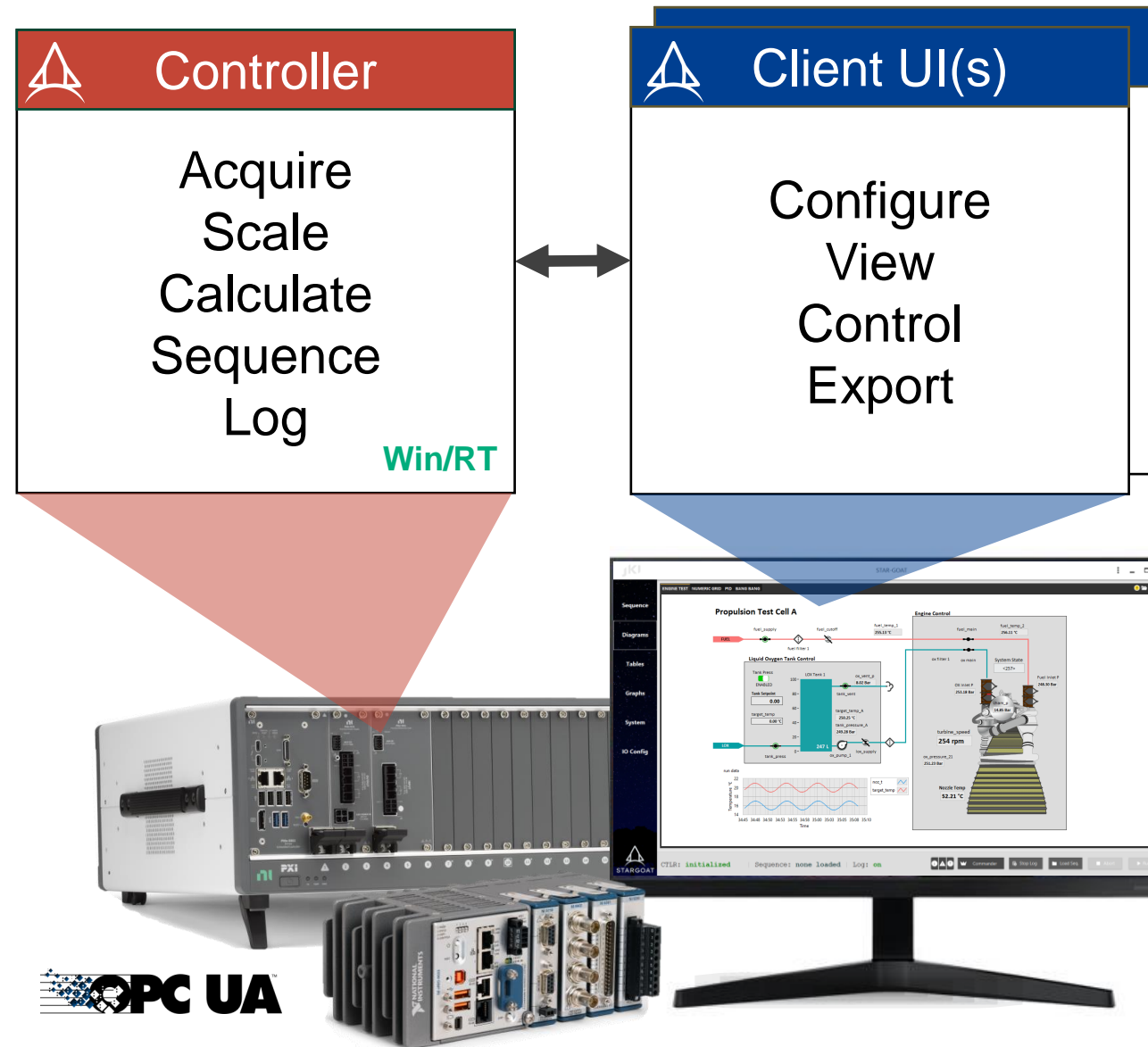




**S**equential **T**ester for **A**erospace **R**esearch

**G**round & **O**peration **A**cceptance **T**esting





## Configure in minutes

- Drag and Drop configuration
- DAQmx (1-100kHz)
- OPC UA
- Customizable multi-step scaling
- Calculated channels

## View Live Data

- Multi-viewer support
- Tables, Pop out plots
- Savable workspaces
- User customizable diagrams

## Deterministic Control

- 1ms sequencing
- Plugin control loops
- PID / FFT / Bang Bang ...



# IO Configuration

## IO

- DAQmx
- OPC UA

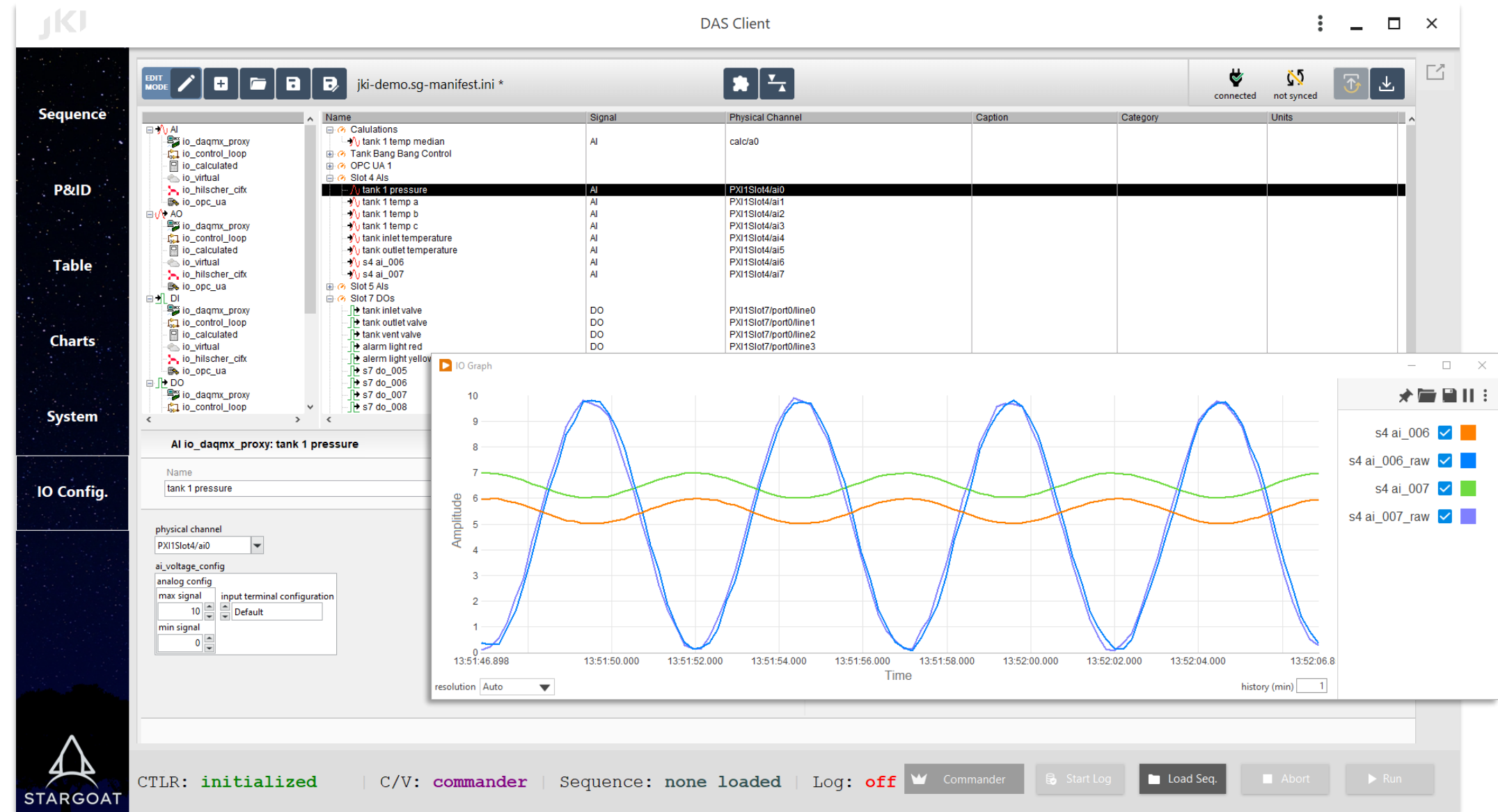
Scaling, taring (zeroing)

Raw & Scaled Data

Calculated Channels

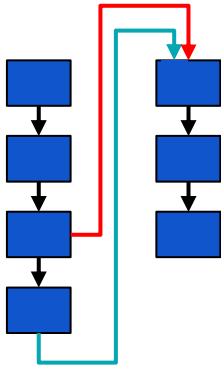
## Control Loops

- PID
- FFT
- Bang Bang





# Sequencing



Sequence

P&ID

Table

Charts

System

IO Config.

STAR-GOAT DAS - Example Sequence

Loaded Sequence

Sequence Name  
Example Sequence

Elapsed Time  
00.00 sec

Subseq time  
00.00 sec

16 steps

Active Conditions

Channel	Min	Max	Rule	Subsequence
(G) System_Abort Sec	1.00	1.00	true if in bounds	Abort
(G) Virtual_Out_002	500.00	Inf	true if in bounds	Abort

Abort Status

Tripped abort conditions

Channel	Min Bound	Max Bound	Rule	Channel value (most recent)
---------	-----------	-----------	------	-----------------------------

SubSequence	Steps	IO	Conditions	Start Time	End Time
Pre-Fire	RAG Light	Virtual_Out_001: 1		0s	10s
Pre-Fire	Countdown			10s	
Hotfire	open pre valve	s7 do_001: 1		0s	1.05s
Hotfire	open lox valve	s7 do_002: 1		1.05s	1.55s
Hotfire	open fuel valve	s7 do_003: 1		1.55s	2s
Hotfire	Step 1	Virtual_Out_001: 1		2s	3s
Hotfire	step 2	Virtual_Out_001: 2		3s	4s
Hotfire	step 3	Virtual_Out_001: 3		4s	5s
Hotfire	step 4	Virtual_Out_001: 4		5s	10s
Hotfire	Shutdown-prep	Virtual_Out_001: 5		10s	15s
Hotfire	End			15s	
Safe	close_lox_valve	Virtual_Out_001: 0		0s	2s
Safe	close_fuel_valve	s7 do_002: 0			
Safe	set_analog_channel	s7 do_001: 0			
Safe	close_other_valve	s7 do_003: 0		2s	
Abort	set analog	s7 do_003: 0		0s	1s
Abort	turn on fire suppression	s7 do_002: 0			
Abort	close lox valve	s7 do_001: 1			
Abort	close fuel valve	Virtual_Out_001: 3.14			
Abort	wee-ooo	s7 do_004: 1		1s	2s
Abort	go to safe			2s	

Pre-Fire

Hotfire

Safe

Abort

CTLR: connected

C/V: commander

Sequence: idle

Log: off

Commander

Start Log.

Clear Seq.

Abort

Run

data\_log file prefix:

High precision output control ~1ms resolution

Conditional monitoring and branching

Automated / Manual Abort

Excel (csv) based config



# Diagrams

View & Command any channel

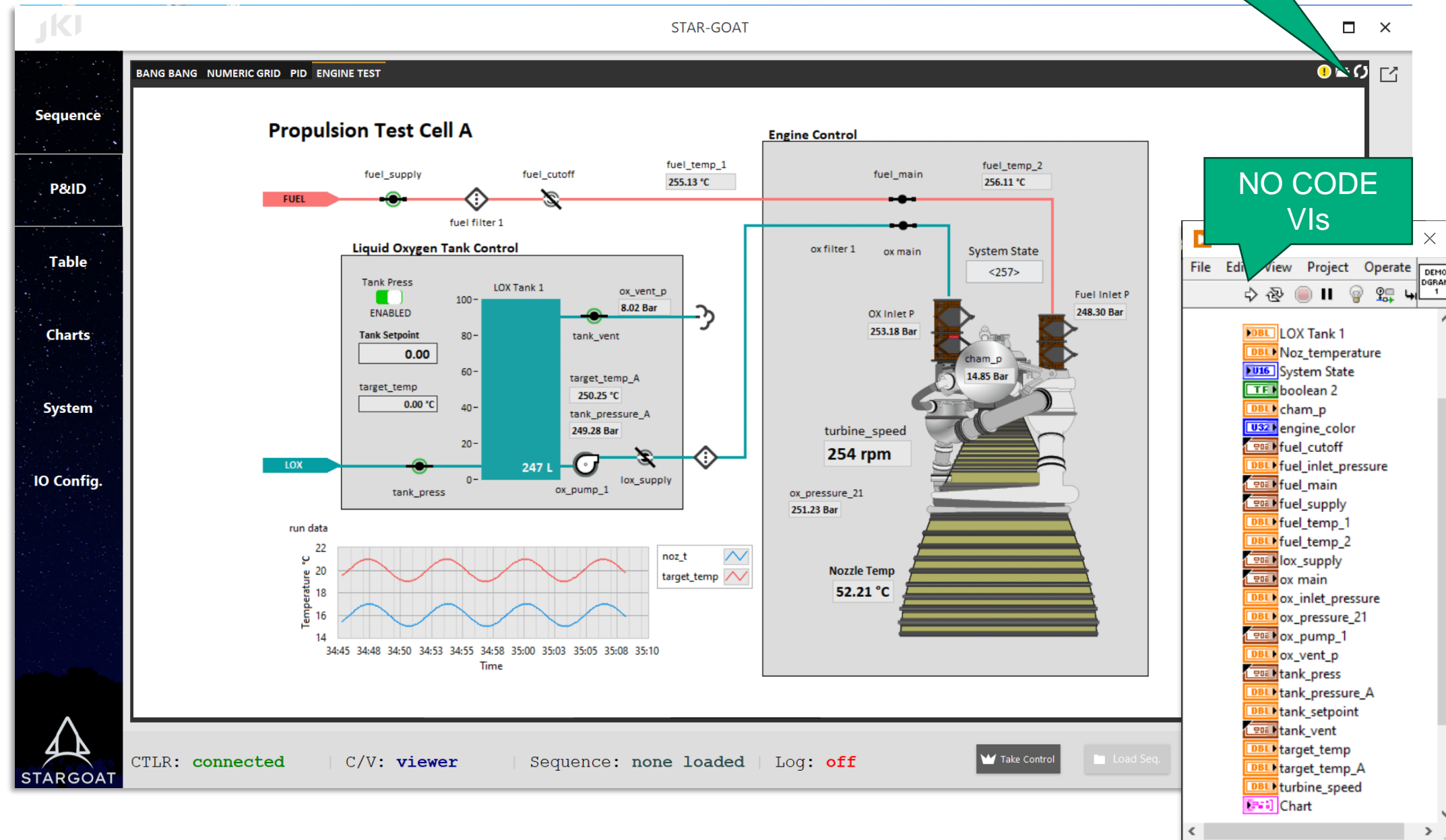
“Codeless” user diagrams

Edit / Reload at runtime

Multi-screen viewing

Reload at Runtime

NO CODE VIs

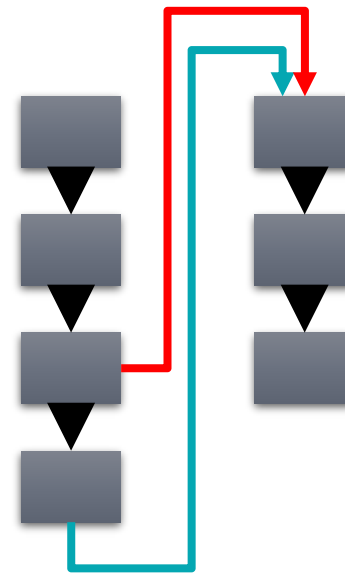




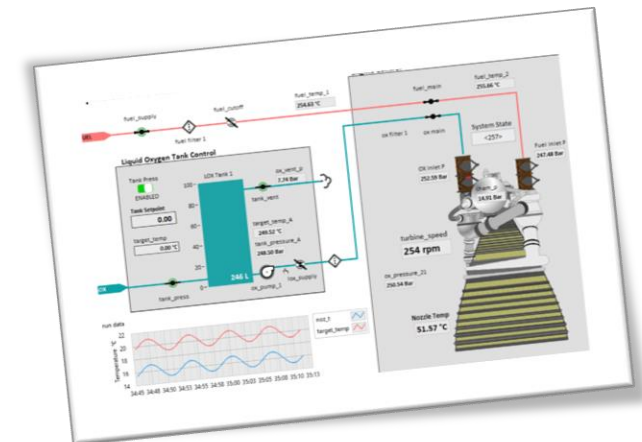
# STAR-GOAT



Hardware



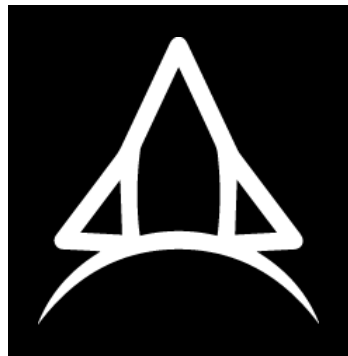
Process



Interface

# JKI | Products

Solve



STAR-GOAT



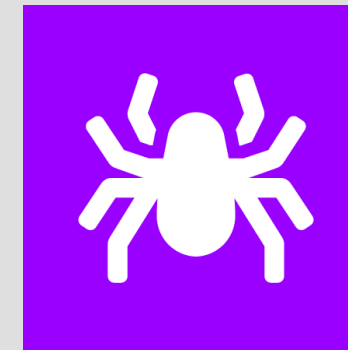
VIPM PRO  
Package Manager



J-Builder  
App Builder



J-Tester  
Unit Testing



J-Crawler  
Static Analysis



Pegasus  
Test Executive

CI/CD Suite



# Check out STAR-GOAT on the Expo Floor



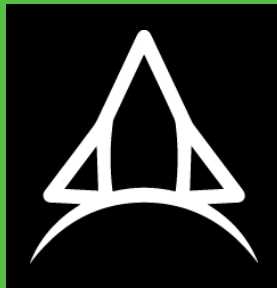




**Hunter Smith**  
Staff Software Engineer  
hunter@jki.net



**Tarek Safwan**  
Sales Account Executive  
tarek@jki.net

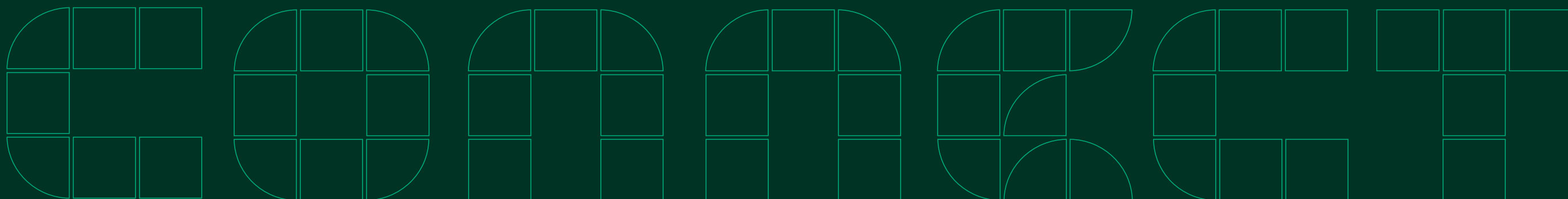


**STAR-GOAT Inquires**  
star-goat@jki.net



**Steve Summers**  
steve.summers@ni.com

[jki.net/star-goat](https://jki.net/star-goat)



**Thank You**



## Functional Requirements



## Assurance Requirements



## Process Requirements

House rules

## jki | Products



CI/CD Suite

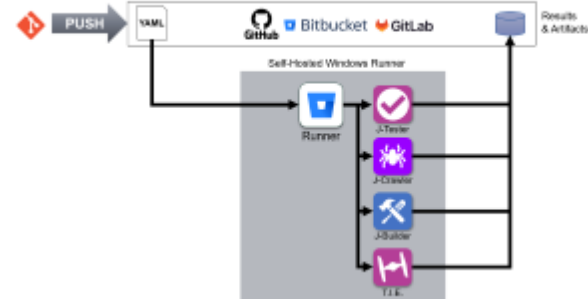
## Hardware

What are we connecting to?

- Channel Count
- Signal Types
- Data Rates
- Industrial Protocols
  - OPC, Modbus, Profinet
- Scaling
- Filtering



## Automate with CI/CD



## SDP

The process behind the process

- Source Controlled
- Generated
- HTML & PDF
- Integrated Metrics



## STAR-GOAT Diagrams

- View & Command any channel
- "Codeless" user diagrams
- Edit / Reload at runtime
- Multi-screen viewing



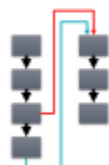
jki.net/star-goat



## Functional Requirements



Hardware



Process

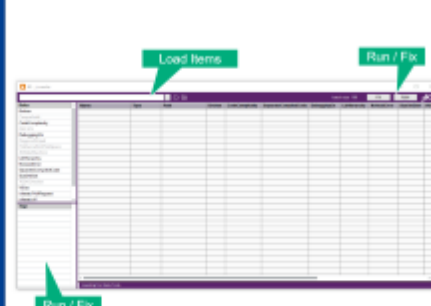


Interface

## J-Crawler

Static Analysis Utility

- Load dir, class, lvproj
- Run "Rules"
- Audit
- Fix
- Beta Available Soon



## SDP

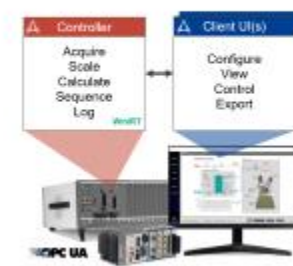
Software Development Plan

- 9000+ words
- 81 SWE Requirements
- 28 CWE Security Policies
- Signed & Tracked
- Binding



- Project Management**
  - Project Roles
  - Developer Training
  - Timelines
  - Estimates
  - Budgets
  - Issue Tracking
  - Issue Workflow
  - Requirement Tracking
  - Audit and Oversight
  - Milestones
- Configuration Management**
  - Software Records
  - Branches and Releases
  - Source Code Control
  - Environmental Configuration
  - DevOps Infrastructure
  - Deployment Configuration
- Software Design**
  - System Architecture
  - COTS Software
  - Software Reusability
  - Software Data
  - Threat Assessment
  - Software Classification
- Development Standards**
  - Performance Design
  - Data Segregation
  - Input Validation
  - Error Handling
  - Fail Safe Design
  - Coding Standards
  - Security Checklists
  - Link Level Memory Access
  - Hardware Communication
  - Encryption
  - FIPS 140
  - Systems Commands
  - Authentication
  - User Testing
  - Code Coverage
  - Software Measurements
  - Performance Metrics
  - State Machine
- Verification and Validation**
  - Deliverables
  - Peer Reviews
  - Pull Requests
  - Developer Workflows
  - Member Checklists
  - Integration Testing
  - Delivery Verification
  - Verification Reports
  - Hardware In The Loop Testing

## STAR-GOAT



- Configure in minutes**
  - Drag and Drop configuration
  - Discrete (1-100k/s)
  - OPC UA
  - Customizable multi-stop scaling
  - Calculated channels
- View Live Data**
  - Multi-viewer support
  - Tables, Pop out plots
  - Savable workspaces
  - User customizable diagrams
- Deterministic Control**
  - True sequencing
  - Plug-in control loops
  - PD / FFT / Bang Bang



jki.net/star-goat